UCLouvain

École polytechnique de Louvain

Nearest correlation matrix with constraints: a problem from finance

Author: **Benjamin CHARLES**
Supervisors: **Pierre-Antoine ABSIL, Antoine VANDENDORPE**
Reader: **Jean-Charles DELVENNE**
Academic year 2022–2023
Master [120] in Mathematical Engineering

Acknowledgements

I would like to take this opportunity to thank all the people who have contributed to this master thesis. Particularly, my two advisors Pierre-Antoine Absil and Antoine Vandendorpe for their support and guidance throughout the thesis. I would also like to thank Jean-Charles Delvenne for accepting to read this master thesis.

Contents

1 Introduction

1.1 Financial context

The financial market consists of two primary segments: **the spot markets** and **the derivatives markets**.

In the **spot markets**, assets are sold and delivered immediately. For example, stocks and bonds are traded in this market. A stock is a type of security that represents a share of ownership in a company and a bond is a debt instrument or security that represents a loan made by an investor to a borrower.

In the **derivatives markets**, financial derivatives are exchanged. The value of these instruments depends upon the price of an other asset, called the underlying. The underlying can be of different types (stocks, bonds, ...). For example, let us consider the European call option, which is a common type of financial derivative. The definition comes from [1].

Definition 1 (European call)**.** *The owner of an European call option on a stock S (the underlying), of exercise date T and strike price K, has the right to buy, at time T, the asset S, at price K. In practice, if S_t represents the price of the underlying at time t, the payoff of the call is $max(S_T - K, 0)$.*

Note that the price on which the option can be sold at time 0 depends on its expected value at time T (i.e. its expected payoff). One important technique used to price such option is to use a Monte-Carlo algorithm. The procedure works as follows :

- Find a stochastic model that well represents the behavior of the stock price S_t. A Brownian motion is often used.
- Simulate a large number of trajectories for the stock price S_t based on the stochastic model developed.
- Find the average payoff received for all the simulated trajectories.
- Based on the average payoff, compute the value of the option at time 0. The price corresponds to the average payoff of the option at time T, actualized by the risk free rate.

Note that there exist financial derivatives that depend on more than one underlying. For example, the best-of call [2] :

Definition 2 (Best-of call)**.** *The owner of a best-of call option on multiple stocks $(S^{(1)}, S^{(2)}, ..., S^{(n)})$ (the underlyings), of exercise date T and strike performance K, has the right to receive, at time T, the highest positive performance achieved, i.e. the payoff of the option is :*

$$\text{payoff} = \max\left[0, \max\left(\text{Perf}(S^{(1)}), \text{Perf}(S^{(2)}), ..., \text{Perf}(S^{(n)})\right) - K\right] \tag{1}$$

with

$$\text{Perf}(S^{(i)}) := \frac{S_T^{(i)}}{S_0^{(i)}} = \frac{\text{Price of stock } S^{(i)} \text{ at time T}}{\text{Price of stock } S^{(i)} \text{ at time 0}}. \tag{2}$$

Even if the problem is more complex, the option can also be priced with a Monte-Carlo algorithm. The additional difficulty comes from the fact that n stock prices have to be modeled (one for each stock price $S_t^{(i)},\ i = 1, ..., n$) and that historical data show that the behaviors of the stock prices are often correlated. The model used to simulate the stock prices should take into account these correlations.

More precisely, let us imagine that we have to simulate the price behaviors of n different stocks $(S^{(1)}, S^{(2)}, ..., S^{(n)})$. All the correlations between the stocks can be gathered in a correlation matrix A constructed as follows :

$$A_{ij} = \text{correlation between stock price } i \text{ and stock price } j \qquad i, j = 1, ..., n. \tag{3}$$

Indeed, since the stock price at a future time is stochastic, it can be represented by a random variable. The correlation associated to two random variables measures the linear relationship between them and is a number contained in $[-1, 1]$. A correlation matrix has the following properties : positive semidefinitiveness, unit diagonal and symmetry.

Using the empirical correlations obtained from the historical data, we can construct an approximate correlation matrix A. Only the positive semidefinitiveness property is not always satisfied when the correlation matrix is empirical.

It is worth noting that some correlations are more significant to price the option accurately. It can be for example the ones that are the most reliable with respect to the historical data obtained. To conclude, what is really needed to price accurately the option is a partial approximate correlation matrix A, i.e. an approximate correlation matrix A where some of the elements (the most important ones) are specified.

In practice, to price the option, it is very important that the partial matrix admits a completion that is positive semidefinite, i.e. that we can fill the unspecified values in A and obtain a symmetric and positive semidefinite matrix. The reason is clear : if the matrix does not admit a positive semidefinite completion, the correlations specified in A are necessarily not coherent and the price obtained from the model will not correspond to the true value of the option.

The objective of this master's thesis, conducted in collaboration with BNP Paribas, is to find an algorithm able to transform A into a partial matrix that admits a semidefinite completion. The mathematical problems are carefully defined in subsection 1.2.

1.2 Mathematical problems

The previous section left us with a partially specified matrix $A \in \mathbb{R}^{n\times n}$. We define an index set Ω that contains the indices where A is specified :

$$\Omega = \{(i,j) \in \{1,...,n\}^2 : A_{ij} \text{ is specified}\}.$$

The requirements for the matrix A are the following :

1. A is symmetric, i.e. A_{ij} specified implies A_{ji} specified and $A_{ij} = A_{ji}$.
2. Each specified value of A represents a correlation and hence is contained in $[-1, 1]$.
3. All the diagonal elements of A are specified and equal to one.

A first problem of interest is to compute the nearest correlation matrix of A, under the constraints that $X_{ij} = A_{ij}$ for all $(i,j) \in \Omega$. The notation $X_\Omega = A_\Omega$ is introduced to represent the constraints in a more convenient manner. For this first problem, the missing entries of the matrix A can be filled with values in $[-1, 1]$.

Let $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n A_{ij}^2}$ denote the Frobenius norm of the matrix A and let $\mathbb{S}^n$ denote the set of symmetric matrices $\in \mathbb{R}^{n\times n}$. The nearest correlation problem with constraints can be written mathematically as follows :

Problem 1 (**Nearest correlation problem with constraints**).

$$\begin{aligned} \min_{X\in\mathbb{S}^n} \quad & \|A - X\|_F^2 \\ s.t. \quad & X_\Omega = A_\Omega, \\ & X \succeq 0. \end{aligned}$$

Several algorithms, using different approaches, have been developed to solve this problem. A first approach is to use the alternating projection algorithm [3]. The main problem with this approach is that the rate of convergence is at best linear. A second approach is to use a shrinking method [4] but with certain limitations as it requires the fixation of an entire leading block in A. In other words, there are conditions on the index set Ω. Finally a third approach based on strongly semismooth functions has been developed to solve the problem [5]. The principal improvement of this method is its quadratically convergence rate.

In practice, it happens that the constraints $X_\Omega = A_\Omega$ make the problem infeasible, meaning that there exists no matrix $X \succeq 0$ such that $X_\Omega = A_\Omega$. Asking whether or not problem 1 is feasible is the same as asking if there exists a positive semidefinite completion of the partial matrix A.

Therefore, a second problem of interest is to find the minimal number of off-diagonal values of A to modify in order to make the problem 1 feasible.
Let us define a function L as follows :

$$L : \mathbb{R}^2 \to \{0,1\} : (x,y) \to \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

This second problem of interest can be written mathematically as follows :

Problem 2.

$$\begin{aligned} \max_{\bar{A}\in\mathbb{S}_n} \quad & \sum_{(i,j)\in\Omega} L(A_{ij},\bar{A}_{ij}) \\ s.t. \quad & \bar{A}_{ii} = 1 \quad i = 1,...,n, \\ & \bar{A} \succeq 0. \end{aligned}$$

Note that the problem 2 is discrete and is always feasible. Indeed, to satisfy the constraints, it is sufficient to consider $\bar{A} = I_n$ (the identity matrix of size $n \times n$).
It turns out that this problem is particularly complicated. According to [6], "the problem of asking whether a partial rational symmetric matrix with an all-ones diagonal can be completed to a full positive semidefinite matrix of rank at most k, is NP-hard for any fixed integer $k \geq 2$". Hence, it is NP-hard when $k = n$, the case of interest.

Let $\circ$ denote the Hadamard product, i.e. $(A \circ B)_{ij} = A_{ij}B_{ij}$, where A and B are two matrices of the same size. The third problem of interest is the following :

Problem 3.

$$\begin{aligned} \min_{\bar{A}\in\mathbb{S}_n} \quad & ||H \circ (A - \bar{A})||_F^2 \\ s.t. \quad & \bar{A}_{ii} = 1 \quad i = 1,...,n, \\ & \bar{A} \succeq 0. \end{aligned}$$

The matrix $H \in \mathbb{S}^n$ is a weight matrix that allows to emphasize the specified elements in A.
For example, the matrix H could be :

$$H_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases} \qquad i,j = 1,...,n. \tag{5}$$

Several algorithms were developed to solve this convex problem ([7], [8]).

It is interesting to notice the difference between problem 2 and 3. In problem 2, we are really interested in modifying the minimal number of off-diagonal elements specified in A, while in problem 3, we can modify all of them but such that the Frobenius norm of the difference of the specified elements is the smallest.

The objective of the master thesis is to find an algorithm that is able to solve problem 2 but by modifying as little as possible the elements specified in A. Let $\beta \in \mathbb{R}^+$ be a parameter. Mathematically, we want to solve the following problem :

Problem 4.

$$\begin{aligned} \max_{\bar{A}\in\mathbb{S}_n} \quad & \sum_{(i,j)\in\Omega} L(A_{ij},\bar{A}_{ij}) - \beta||H\circ(A-\bar{A})||_F^2 \\ s.t. \quad & \bar{A}_{ii} = 1 \quad i = 1,...,n, \\ & \bar{A} \succeq 0. \end{aligned}$$

The algorithm should be designed to perform well with inputs (partial matrices) that are approximately of the same size as the one provided by BNP.

In subsection 1.3, a brief explanation of how problem 2 can be interpreted geometrically is made.

In section 2, we explore a first approach to solve problem 4. The approach relies on semidefinite optimization and on the strong duality theorem. More precisely, we formulate a semidefinite optimization problem that gives information about the feasibility of problem 1. We use the dual variables of this problem to obtain information about the elements in A that are the most relevant to modify in order to obtain a feasible problem. We will observe that the primary limitation of this approach lies in its high computational cost.

In section 3, a second approach to solve problem 4 is made. The main objective of this approach is to be less computationally expensive than the previous one. This approach is based on the observation that a partial matrix can be represented by a graph and that the structure of the graph has a strong impact on the positive semidefinite completion problem. The particular structure of the graph representing the partial matrix provided by BNP will be exploited to obtain a more efficient algorithm.

1.3 Geometric point of view of the problem

From a geometrical point of view, the problem of semidefinite completion can be seen as asking whether or not there exists a point in the intersection between the cone of semidefinite matrices $\mathbb{S}_n^+$ and the affine subspace defined by the linear constraints specified in A_Ω. If we include the diagonal constraints directly in $\mathbb{S}_n^+$, the cone of semidefinite matrices becomes an elliptope. The next definition comes from [9].

Definition 3 (Elliptope)**.** *The set of all real, square symmetric matrices whose diagonal entries are all equal to one and whose eigenvalues are all non-negative (also called the set of correlations matrices).*

The 3-dimensional elliptope defined by the set :

$$\left\{ (x, y, z) \in \mathbb{R}^3 : \begin{bmatrix} 1 & x & y \\ x & 1 & z \\ y & z & 1 \end{bmatrix} \succeq 0 \right\} \tag{6}$$

is represented on the left graph of figure 1. Let us imagine that the two constraints in this small problem are $x = 0.8$ and $y = 0.6$. The affine subspace represented by the linear constraints is therefore the line intersecting the planes of equation $x = 0.8$ and $y = 0.6$. The intersection of the elliptope and this line gives all the solutions of the semidefinite completion problem, as represented on the right graph of figure 1.

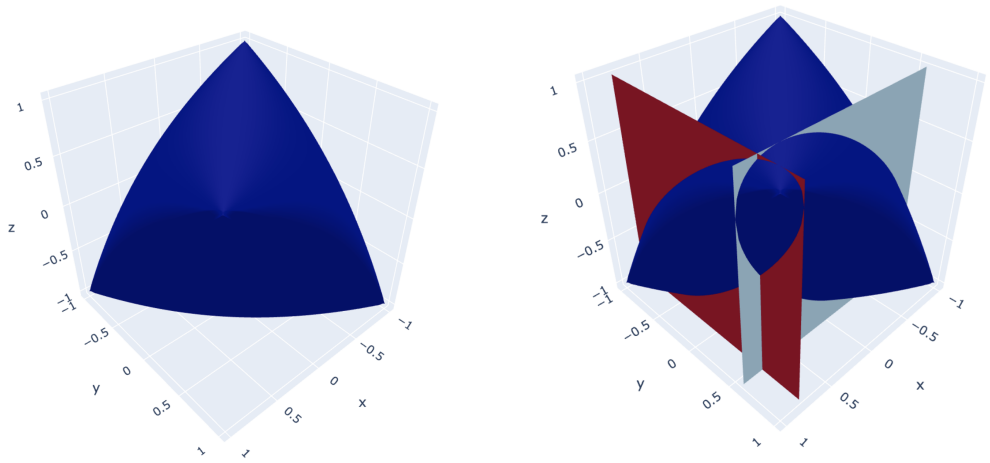

Figure 1: **Left** : 3D elliptope. **Right** : 3D elliptope and affine subspace.

One approach for solving problem 2 would be to find the constraint(s) in the affine subspace that make(s) the semidefinite completion problem infeasible, i.e. the constraint(s) that lead(s) to an empty intersection between the elliptope and the affine subspace.

The geometrical approach was not explored here due to the difficulty to generalize the problem in higher dimensions.

2 First approach : Semidefinite optimization and duality

2.1 Preliminaries

In this section, we remind what a semidefinite optimization problem is and where it comes from. We first have to briefly recall basic concepts of linear and conic optimization.

Let A be a matrix $\in \mathbb{R}^{n\times m}$, b a vector $\in \mathbb{R}^m$ and c a vector $\in \mathbb{R}^n$. The linear optimization problems can be written in primal standard form as follows :

Problem 5.

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x \\ s.t. \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

This problem admits the following dual problem :

Problem 6.

$$\begin{aligned} \max_{y\in\mathbb{R}^m} \quad & b^T y \\ s.t. \quad & A^T y \leq c. \end{aligned}$$

It is well known that the two following theorems (taken from [10]) are verified for the linear optimization problems.

Theorem 1 (Weak duality)**.** *The objective value of any feasible solution of the dual problem is an upper bound on any feasible solution of the primal problem.*

Theorem 2 (Strong duality)**.** *The objective value of the optimal solution of the primal problem is equal to the objective value of the optimal solution of the dual problem.*

The goal of the conic optimization is to generalize the linear optimization while trying to keep the properties of weak and strong duality. The strong duality theorem is very useful to perform a sensitivity analysis, i.e. to analyze how changes in the parameters of the problem affect the optimal solution.

In order to generalize the linear problems, we replace the inequality $x \geq 0$ in problem 5 by $x \succeq_K 0$, which means $x \in K$ (with K a convex cone). We define a partial order :

$$x \succeq_K a \iff x - a \succeq_K 0 \iff x - a \in K. \tag{7}$$

In order for the weak duality to hold in conic optimization, the inequality in the dual problem 6 becomes :

$$A^T y \preceq_{K^*} c \qquad \text{where } K^* := \{z \in \mathbb{R}^n \text{ such that } x^T z \geq 0 \ \forall x \in K\}. \tag{8}$$

The cone K^* is called the dual cone of K.

It is straightforward to show that the weak duality theorem holds for the conic optimization problems:

Proof. $c^T x - b^T y = x^T c - (Ax)^T y = x^T c - x^T A^T y = \underbrace{x^T}_{\in K} \underbrace{(c - A^T y)}_{\in K^*} \geq 0.$ □

A feasible solution to a conic optimization problem is strictly feasible if it belongs to the interior of the cone, i.e :

$$Ax = b \text{ and } x \succ_K 0 \quad \text{for the primal,} \tag{9}$$
$$A^T y \prec_{K^*} c \qquad \text{for the dual.} \tag{10}$$

It is well-known that the strong duality does not always hold in conic optimization. However, the following theorem provides a sufficient condition to have strong duality in conic optimization.

Theorem 3 (Strong duality in conic optimization)**.** *If the primal problem admits a strictly feasible solution and if the primal is bounded (i.e. the optimal value does not tend to $-\infty$), then the dual is solvable and both optimal values are equal, i.e. the strong duality holds. The theorem stays true if we exchange dual and primal.*

Proof. A proof of this theorem is available in [11]. □

Semidefinite optimization problems are conic optimization problems restricted to the semidefinite cone $\mathbb{S}^n_+$, i.e. the cone of the positive semidefinite matrices. With that cone, we obtain the Löwner order $\succeq_{\mathbb{S}^n_+}$:

$$A \succeq_{\mathbb{S}^n_+} 0 \quad \text{if and only if } A \text{ is positive semidefinite.}$$
$$A \succeq_{\mathbb{S}^n_+} B \quad \text{if and only if } A - B \text{ is positive semidefinite.}$$

Note that to facilitate the notation and when the context is clear, we replace $\succeq_{\mathbb{S}^n_+}$ by $\succeq$.

The only thing left is to adapt the conic formulation to work with matrix variables. First, we define the following inner product between two matrices :

$$\langle S, T \rangle := \text{Tr}(S^T T) = \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij} T_{ij} := S \bullet T. \tag{11}$$

The corresponding induced norm is the Frobenius norm :

$$S \bullet S := ||S||_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}^2. \tag{12}$$

We can show that with the inner product defined in equation 11, the semidefinite cone is self dual, meaning that $(\mathbb{S}^n_+)^* = \mathbb{S}^n_+$.

With the notations introduced, the primal conic problem with the semidefinite cone can be written as :

Problem 7.

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \quad & C \bullet X \\ s.t. \quad & A_i \bullet X = b_i \quad i = 1, ..., m, \\ & X \succeq 0. \end{aligned} \tag{13}$$

The matrices $A_i, C \in \mathbb{S}^n$ and the scalars $b_i \in \mathbb{R}$ represent the problem data.
The dual of this problem is the following :

Problem 8.

$$\begin{aligned} \max_{y \in \mathbb{R}^m} \quad & b^T y \\ s.t. \quad & \sum_{i=1}^{m} A_i y_i \preceq C. \end{aligned} \tag{14}$$

2.2 Semidefinite optimization problem

Let us define the matrix $E_{(i,j)} \in \mathbb{R}^{n\times n}$ by a matrix full of zeros, except at the index (i,j) and (j,i), where it is equal to $\frac{1}{2}$ if $i \neq j$, or equal to 1 if $i = j$.

For example, if $n = 3$, we have :

$$E_{(1,1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad E_{(1,2)} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{15}$$

Let us define $\bar{\Omega}$ by:

$$\bar{\Omega} = \{(i,j) \in \Omega \text{ s.t. } i \leq j\} \tag{16}$$

and let I_n denote the identity matrix of size n.
The following semidefinite optimization problem is particularly useful to gain insight into the feasibility of problem 1.

Problem 9.

$$\begin{aligned} \min_{X\in\mathbb{S}^n, s\in\mathbb{R}} \quad & s \\ s.t. \quad & X + sI_n \succeq 0, \\ & E_{(i,j)} \bullet X = A_{ij} \quad \forall (i,j) \in \bar{\Omega}. \end{aligned}$$

Indeed, the constraints $E_{(i,j)} \bullet X = A_{ij} \quad \forall (i,j) \in \bar{\Omega}$ are simply a reformulation of the constraints $X_\Omega = A_\Omega$. The problem 1 is feasible if the optimal value of problem 9 is smaller than zero.

Note that problem 9 is not written in the exact same form as problem 7. In fact, it is easier to formulate problem 9 under the dual form directly.

Let A_F be a completion of the partial matrix A with zeros and let $\hat{\Omega}$ be the set defined by:

$$\hat{\Omega} = \{(i,j) \in \{1,...,n\}^2, i < j, (i,j) \notin \bar{\Omega}\}. \tag{17}$$

The problem 9 can be written as :

Problem 10.

$$\begin{aligned} -\max_{x_{ij}, s\in\mathbb{R}} \quad & -s \\ s.t. \quad & \sum_{(i,j)\in\hat{\Omega}} 2E_{(i,j)}x_{ij} + I_n s + A_F \succeq 0. \end{aligned}$$

The dual of this problem is :

Problem 11.

$$\begin{aligned} -\min_{Y\in\mathbb{S}^n} \quad & A_F \bullet Y \\ s.t. \quad & -2E_{(i,j)} \bullet Y = 0 \quad \forall (i,j) \in \hat{\Omega}, \\ & -I_n \bullet Y = -1, \\ & Y \succeq 0. \end{aligned}$$

Example :

Consider the following partial matrix A :

$$A = \begin{pmatrix} 1 & - & - \\ - & 1 & 0.5 \\ - & 0.5 & 1 \end{pmatrix}.$$

We have :

$$\begin{aligned} \Omega &= \{(1,1),(2,2),(3,3),(2,3),(3,2)\}, \\ \bar{\Omega} &= \{(1,1),(2,2),(3,3),(2,3)\}, \\ \hat{\Omega} &= \{(1,2),(1,3)\}. \end{aligned}$$

The problem 10 can be written as follows :

$$\begin{aligned} -\max_{x_{12},x_{13},s\in\mathbb{R}} \quad & -s \\ \text{s.t.} \quad & -\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x_{12} - \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} x_{13} - \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} s \preceq \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix}. \end{aligned}$$

The corresponding dual is :

$$\begin{aligned} -\min_{Y\in\mathbb{S}^3} \quad & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix} \bullet \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{pmatrix} \\ \text{s.t.} \quad & -\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \bullet \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{pmatrix} = 0, \\ & -\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \bullet \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{pmatrix} = 0, \\ & -\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{pmatrix} = -1, \\ & \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{pmatrix} \succeq 0. \end{aligned}$$

Note that since Y is symmetric, the first and second constraints impose respectively that $Y_{12} = Y_{21} = 0$ and $Y_{13} = Y_{31} = 0$. The third constraint imposes that $Y_{11} + Y_{22} + Y_{33} = 1$. Taking that into account, the objective function of the dual problem becomes :

$$-\min_{Y \in \mathbb{S}^3} \quad 1 + 2A_{F_{23}}Y_{23} \qquad \text{where } A_{F_{23}} = A_{23} = 0.5$$

This last observation can be generalized for any problems of type 11 :

Proposition 1. *The objective function of problem 11 can be written as follows :*

$$-\min_{Y \in \mathbb{S}^3} \quad 1 + \sum_{(i,j) \in \bar{\Omega}, i \neq j} 2A_{ij}Y_{ij}.$$

Proof. It is a direct consequence of the constraints. □

If the strong duality holds for the problem 10, this last proposition gives us useful information about the constraints specified in A that have an impact on the objective function. Indeed, the dual variables give the sensitivity of the objective function with respect to the values specified in A.

To make it more formal, let $V^*(A)$ and Y be respectively the optimal value and the matrix of dual variables obtained when solving problem 11 for the partial matrix A. Suppose that Y_{kl} $(k \neq l)$ is the largest dual variable in absolute value. Let $\epsilon \in \mathbb{R}^+$ and let us define the partial matrix B as follows :

$$B_{ij} = \begin{cases} A_{ij} & \text{If } (i,j) \neq (k,l) \text{ or } (i,j) \neq (l,k) \\ A_{ij} + \epsilon \times \text{sign}(Y_{kl}) & \text{Otherwise} \end{cases} \qquad (i,j) \in \Omega.$$

Let's also imagine that the matrix of dual variables Y of the optimization problem does not change for the partial matrices A and B. We have the following relation :

$$\begin{aligned} V^*(B) &= -B_F \bullet Y && (18) \\ &= -(A_F \bullet Y + 2\epsilon \times \text{sign}(Y_{kl})Y_{kl}) && (19) \\ &= V^*(A) - 2\epsilon \times |Y_{kl}|. && (20) \end{aligned}$$

If the strong duality holds and if the matrix Y stays optimal for small changes in A, the last result shows that the variable s in problem 9 decreases when passing from the partial matrix A to the partial matrix B.

If the optimal solution Y changes to $\tilde{Y}$ when we pass from A to B, we obtain the following inequalities:

$$\begin{aligned} & B_F \bullet \tilde{Y} \leq B_F \bullet Y = A_F \bullet Y + 2\epsilon|Y_{kl}| && (21) \\ & \Rightarrow V^*(B) \geq V^*(A) - 2\epsilon|Y_{kl}| && (22) \\ & \Rightarrow V^*(A) - V^*(B) \leq 2\epsilon|Y_{kl}|. && (23) \end{aligned}$$

The inequality 21 is justified by the fact that if Y is a feasible solution of problem 11 for A, then it remains feasible for B. Hence, we necessarily have inequality 21 since it is a minimization problem.

The inequality 23 gives an upper bound on the variation of the optimal value but does not give any guarantees of decrease.

We conclude this subsection by proving that the strong duality holds for problems 10 and 11. We first need the following theorem :

Theorem 4. *A symmetric matrix $A \in \mathbb{S}^n$ that is diagonally dominant and that has real non-negative diagonal entries is positive semidefinite, i.e. :*

$$A_{ii} \geq \sum_{j=1, j\neq i}^{n} |A_{ij}| \quad \forall i \in \{1, ..., n\} \Rightarrow A \succeq 0.$$

If moreover the matrix is strictly diagonally dominant (i.e. we have a strict inequality), then the matrix is positive definite.

Proof. A proof is available in [12]. □

Proposition 2. *The strong duality theorem (theorem 3) holds for problems 10 and 11.*

Proof. We first prove that the problem 10 admits a strictly feasible solution. Let all the variables x_{ij} with $(i, j) \in \hat{\Omega}$, be equal to 0 and let ϵ be a small positive number. Let us define s by :

$$s = \max_{i=1,...,n} \sum_{j=1, j\neq i}^{n} |A_{F_{ij}}| - 1 + \epsilon$$

By theorem 4, we have $I_n s + A_F \succ 0$ since the matrix is strictly diagonally dominant. Moreover, the value of s cannot tend to $-\infty$ since the diagonal elements of the matrix $\sum_{(i,j)\in\hat{\Omega}} 2E_{(i,j)}x_{ij} + I_n s + A_F$ are $(1 + s)$, which implies that s cannot be smaller than 1 (since the matrix must be positive semidefinite). □

2.3 Heuristic

In the section 2.2, we showed that the dual variables of the problem 10 give useful information about the elements in A that have an impact on the objective function. We also showed that slightly modifying the constraint corresponding to the largest dual variable would certainly (if the dual variables do not change too much) result in a decrease of the objective value of problem 10.

In this section, we exploit a smarter approach to modify the constraints in A. The approach also makes use of the dual variables and is based on theorem 5. We first need the following definition :

Definition 4 (Principal minor)**.** *A principal minor of a symmetric matrix A is the determinant of a principal submatrix obtained by removing some rows and the corresponding columns of A.*

Note that we can also define the notion of principal minor for a partial matrix. The only requirement is that the principal submatrix, obtained by removing some rows and the corresponding columns, must be fully specified. All the principal minors of a partial matrix are called the specified principal minors.

Theorem 5. *The symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive semidefinite if and only if all the principal minors of A are nonnegative, or equivalently, if and only if all the principal submatrices of A are positive semi-definite.*

Proof. A proof is available here [12]. □

It turns out that the dual variables often give information about the principal minors of the matrix A that are negative. Consider the following example of partial matrix A :

$$A = \begin{pmatrix} 1 & 0.1 & - & - & 0.5 \\ 0.1 & 1 & - & - & - \\ - & - & 1 & 0.9 & 0.8 \\ - & - & 0.9 & 1 & -0.5 \\ 0.5 & - & 0.8 & -0.5 & 1 \end{pmatrix}. \tag{24}$$

The principal minor obtained by keeping rows 3,4 and 5 and the corresponding columns, is equal to :

$$\det(A_{\{3,4,5\}}) = \det \begin{pmatrix} 1 & 0.9 & 0.8 \\ 0.9 & 1 & -0.5 \\ 0.8 & -0.5 & 1 \end{pmatrix} = -1.42. \tag{25}$$

The notation $A_{\{3,4,5\}}$ represents the submatrix of A obtained by keeping rows 3,4 and 5 and the corresponding columns.

The dual variables of problem 10 obtained with this matrix A as input are :

$$Y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.398 & -0.356 & -0.336 \\ 0 & 0 & -0.356 & 0.318 & 0.300 \\ 0 & 0 & -0.336 & 0.300 & 0.283 \end{pmatrix}. \tag{26}$$

As we can see, the dual variables in Y that are different from zero correspond to the elements in A that lead to a negative principal minor. The largest dual variable in absolute value that is not in the diagonal of Y is -0.356 and is associated to the constraints $A_{34} = 0.9$. Now, instead of reducing A_{34} and A_{43} by a predetermined value ϵ, we decrease them such that the principal minor becomes nonnegative. To find this value, we replace A_{34} and A_{43} by x and compute the value of x such that the principal minor is nonnegative :

$$\det(A_{\{3,4,5\}}) = \det \begin{pmatrix} 1 & x & 0.8 \\ x & 1 & -0.5 \\ 0.8 & -0.5 & 1 \end{pmatrix} = -x^2 - 0.8x + 0.11 \tag{27}$$

$$\det(A_{\{3,4,5\}}) \geq 0 \iff x \in [-0.919, 0.119]. \tag{28}$$

Finally, we replace $A_{34} = A_{43} = 0.9$ by its closest value, i.e. by 0.119.

The procedure can be generalized as follows :

1. Find the matrix of dual variables Y associated to the partial matrix A and take the largest dual variable in absolute value $Y_{kl} \in Y$ with $k \neq l$.

2. Create the following partial matrix :

$$\hat{Y}_{ij} = \begin{cases} Y_{ij} & \text{if } Y_{ij} \neq 0 \\ \text{Not specified} & \text{otherwise} \end{cases} \qquad i, j = 1, ..., n. \tag{29}$$

   Note that $\hat{Y}_{kl}$ is necessarily specified.

3. Find all the principal submatrices of $\hat{Y}$ that are fully specified and that contain $\hat{Y}_{kl}$. In the example considered, the only fully specified submatrix of $\hat{Y}$ that contains $\hat{Y}_{kl} = \hat{Y}_{34}$ is :

$$\hat{Y}_{\{3,4,5\}} = \begin{pmatrix} 0.398 & -0.356 & -0.336 \\ -0.356 & 0.318 & 0.300 \\ -0.336 & 0.300 & 0.283 \end{pmatrix}. \tag{30}$$

   Note that $\hat{Y}_{\{3,4\}}$ is also a fully specified submatrix that contains $\hat{Y}_{34}$ but as it is included in $\hat{Y}_{\{3,4,5\}}$, we do not take it into account. In other words, we only consider the fully specified principal submatrices that are maximal.

   A fully specified principal submatrix in $\hat{Y}$ corresponds necessarily with one fully specified submatrix in A (because $\hat{Y}_{ij}$ specified implies A_{ij} specified). In other words, if $\hat{Y}_{\{3,4,5\}}$ is fully specified, then $A_{\{3,4,5\}}$ is necessarily fully specified.

4. Find if one of the fully specified principal submatrix in A found in step 3 has a negative determinant. This is the case in the example considered for $A_{\{3,4,5\}}$.

5. Replace A_{kl} and A_{lk} by x in the submatrix found in step 4 and look for the closest value of x that gives a nonnegative determinant.

Note that there is no guarantee that steps 4 and 5 are always verified. If it is not the case, we perform a simple update of A_{kl} and A_{lk} by a factor ϵ, as explained in subsection 2.2.

2.4 Algorithm

As a summary, the pseudocode of the algorithm is given in 1. The function `SolveProblemOpti` solves the optimization problem 11 and returns the objective value. In practice, an interior-point type optimizer from the package MOSEK is used to solve the problem 11. The interior point optimizer is an implementation of the homogeneous and self-dual algorithm. A detailed explanation of the algorithm is available here [13]. The package MOSEK is originally coded in C but an interface in Python is available [14].

Algorithm 1:

Function Al1 $(A, \epsilon, \text{Variant})$;

Input : A partial matrix A, a positive number ϵ and a boolean parameter called Variant.
Output: New partial matrix $\bar{A}$ that admits a semidefinite completion.

if *SolveProblemOpti*$(A) \leq 0$ **then**
  $\bar{A} \leftarrow A$
  Return $\bar{A}$
end
while *SolveProblemOpti*$(A) > 0$ **do**
  Find Y_{kl} $(k \neq l)$, the largest dual variable in absolute value.
  if *Variant* = *True* **then**
    Perform update of A_{kl} and A_{lk} as explained in section 2.3.
  end
  else
    $A_{kl} \leftarrow A_{kl} + \epsilon \times \text{sign}(Y_{kl})$
    $A_{lk} \leftarrow A_{lk} + \epsilon \times \text{sign}(Y_{lk})$
  end
end
$\bar{A} \leftarrow A$
Return $\bar{A}$

2.5 Numerical simulations

As explained in the introduction, the objective is that the algorithm performs well on data provided by BNP Paribas. Unfortunately, the number of data provided is limited and not sufficient to test the general behaviour of the algorithm. In this section, we explain how we constructed partially specified matrices and we show the results obtained by applying the algorithm on them.

Let $C \in \mathbb{R}^{n \times n}$ be a random correlation matrix generated by the module RandomCorrMat available in python. Let α be a parameter in $]0, 1[$ and let E be a symmetric matrix of the same size as C and with each entry distributed uniformly $\in [-1, 1]$. We compute the matrix D as follows :

$$D = (1 - \alpha)C + \alpha E. \tag{31}$$

A percentage p of the elements of D are fixed to form the partial matrix A. More precisely, we select randomly $p(n-1)n/2$ elements in the upper triangular part of D (diagonal excluded) and their corresponding elements in the lower triangular part. All the diagonal elements of A are set to one.

To make an informed comparison, the following criteria are considered :

- The number of iterations of the algorithm, i.e. the number of times that the function `SolveProblemOpti` was called.
- The number of elements in the upper triangular part of A that were modified.
- The execution time of the algorithm.
- The value of the following function f:

$$f(A) = ||H \circ (A - \bar{A})||_F^2. \tag{32}$$

  The matrix H is a weight matrix constructed as in equation 5 and $\bar{A}$ is the output of the algorithm obtained with the partial matrix A as input. This function gives information about how much the partial matrix was modified with the algorithm.

- The value of the following function g:

$$g(A) = \frac{\sum_{(i,j) \in \bar{\Omega}, i \neq j} |A_{ij} - \bar{A}_{ij}|}{\sum_{(i,j) \in \bar{\Omega}, i \neq j} |A_{ij}|}. \tag{33}$$

  This function is another way to measure how much the partial matrix was modified with the algorithm.

To compare the solutions obtained with the algorithm (called `Al1`), it is useful to compute the solution of problem 3. The algorithm explained in [7] and implemented in the package NAG [15] was used to solve the problem. As problem 3 is convex, the value of the function $f(A)$ obtained should be minimal. The algorithm is called `Corrmat_h_weight`.

In the rest of this subsection, the parameter α in equation 31 is fixed to 0.8. With such value, there is a very low probability that the partial matrix obtained from D admits a semidefinite completion without any modification (even for a small n and p).

The table 1 contains the numerical results obtained when the percentage p of the elements fixed in the matrix is equal to 0.12 and when the size of the matrix n increases. The third column of the table indicates the algorithm used (`Al1` or `Corrmat_h_weight`) and the parameters of the algorithm. The parameters for algorithm `Al1` are Variant (V) = True (T) or False (F) and the value of ϵ. The other columns correspond to the criteria used to compare the outputs. The results obtained are also represented graphically in figure 2. The graphs compare the values obtained for the different parameters of algorithm `Al1`. The graphs in figure 3 represent, for each n, the value of the function `SolveProblemOpti`, i.e. the optimal value of problem 11, as a function of the number of iterations of algorithm `Al1`.

In the second simulation, the size of the matrix n is fixed to 75 but the percentage p of fixed elements increases. As before, the numerical results obtained are written in table 2 and are represented graphically in figure 4. The graphs in figure 5 represent, for each p, the value of the function `SolveProblemOpti` as a function of the number of iterations of algorithm `Al1`.

Based on the results obtained, we can make several comments :

- As expected, every criteria considered increase with the size of the matrix or with the percentage of fixed elements.
- The first and second graphs of figure 2 and 4 show that the time of execution of algorithm `Al1` is directly related to the number of iterations. We also observe on these graphs that the cost of an iteration (i.e. the time used) increases significantly with the size of the matrix but is relatively independent from the percentage of fixed elements.
- The first and third graphs of figure 2 and 4 show that the number of iterations is not strongly related to the number of modified elements. It means that some elements are modified several times.
- It is useful to compare the strategy (V = T, $\epsilon = 0.05$) and (V = F, $\epsilon = 0.05$) because it shows the impact of the heuristic on the results. We observe that the use of the heuristic generally leads to a smaller number of iterations, to a smaller number of modified elements and to a smaller execution time. However, it leads to a larger value of $f(A)$ and to a similar value of $g(A)$. This last observation is explained by the fact that the use of the heuristic generally modifies the elements in A more importantly, which implies a bigger $f(A)$ since the modifications are raised to the square.
- The graphs on figure 3 and 5 show that the optimal value of problem 11 (i.e. the value of the function `SolveProblemOpti`) seems to decrease at each iteration and for each ϵ considered. We show in subsection 2.6 that it is not always the case.
- As expected, algorithm `corr_h_weight` always gives a value of $f(A)$ smaller than the values obtained with algorithm `Al1`. However, the values of $g(A)$ obtained are often larger! The reason is that algorithm `corr_h_weight` slightly modifies nearly all the elements in A, while algorithm `Al1` significantly modifies only a few elements (resulting in a bigger value of $f(A)$ since the modifications are raised to the square). Note that, when n is sufficiently big, the execution times of algorithm `corr_h_weight` are much lower than the ones obtained with algorithm `Al1`.

p	n	Algorithm and parameters	# of iterations	# of modified elements	Time [s]	$f(A)$	$g(A)$
0.12	75	`All`, (V = T, $\epsilon = 0.05$)	2	1	1.728	0.0431	0.00171
		`All`, (V = F, $\epsilon = 0.02$)	9	1	6.704	0.0512	0.00186
		`All`, (V = F, $\epsilon = 0.05$)	4	1	3.289	0.0450	0.00174
		`All`, (V = F, $\epsilon = 0.1$)	3	1	2.387	0.0800	0.00233
		`corrmat_h_weight`	/	333	1.000	0.0171	0.00195
	100	`All`, (V = T, $\epsilon = 0.05$)	4	3	14.567	0.287	0.00394
		`All`, (V = F, $\epsilon = 0.02$)	33	6	123.734	0.206	0.00413
		`All`, (V = F, $\epsilon = 0.05$)	15	6	55.192	0.240	0.00452
		`All`, (V = F, $\epsilon = 0.1$)	9	6	32.028	0.240	0.00516
		`corrmat_h_weight`	/	594	4.814	0.105	0.00442
	125	`All`, (V = T, $\epsilon = 0.05$)	27	19	437.652	1.310	0.00950
		`All`, (V = F, $\epsilon = 0.02$)	112	25	1825.156	0.722	0.00892
		`All`, (V = F, $\epsilon = 0.05$)	47	21	742.915	0.700	0.00924
		`All`, (V = F, $\epsilon = 0.1$)	26	19	389.701	0.820	0.01004
		`corrmat_h_weight`	/	930	45.029	0.307	0.0161

Table 1: Results obtained when $p = 12\%$, $\alpha = 0.8$.

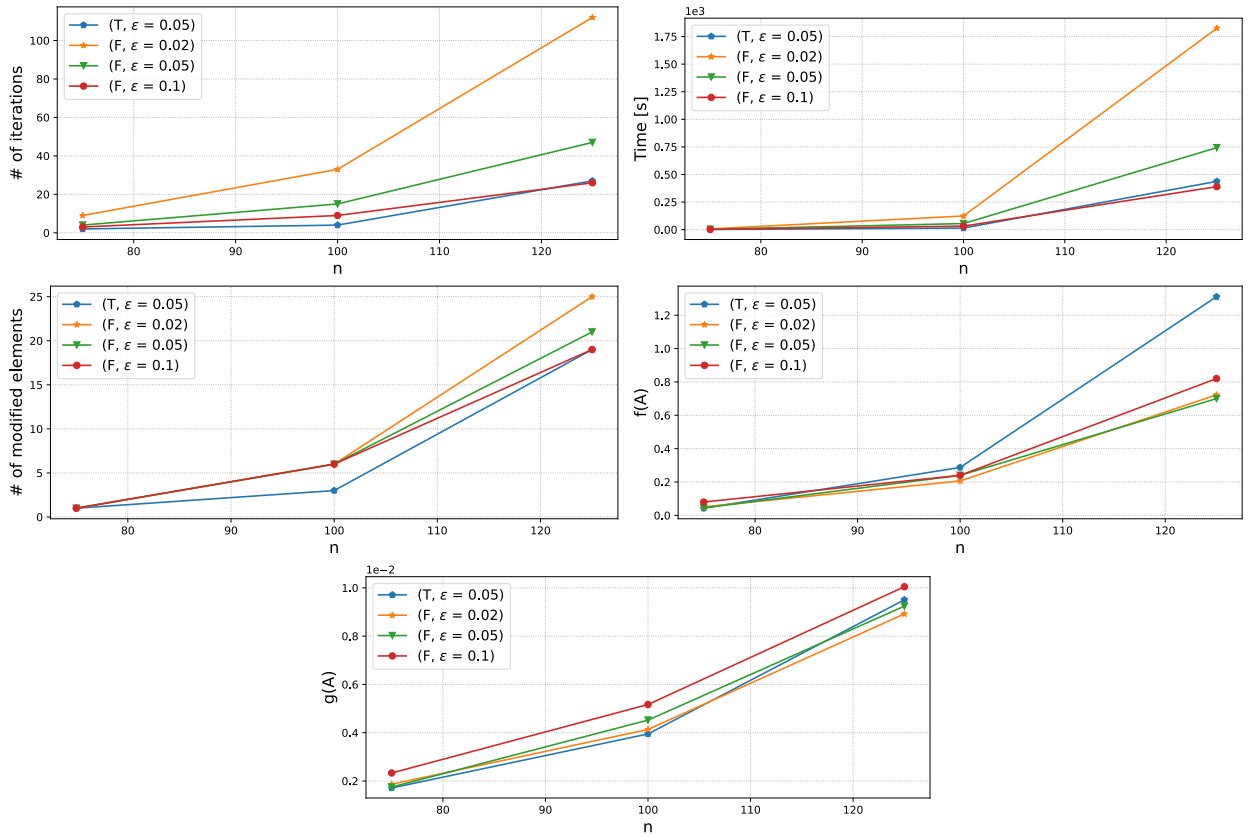

Figure 2: Results obtained with algorithm `All`, $p = 12\%$, $\alpha = 0.8$.

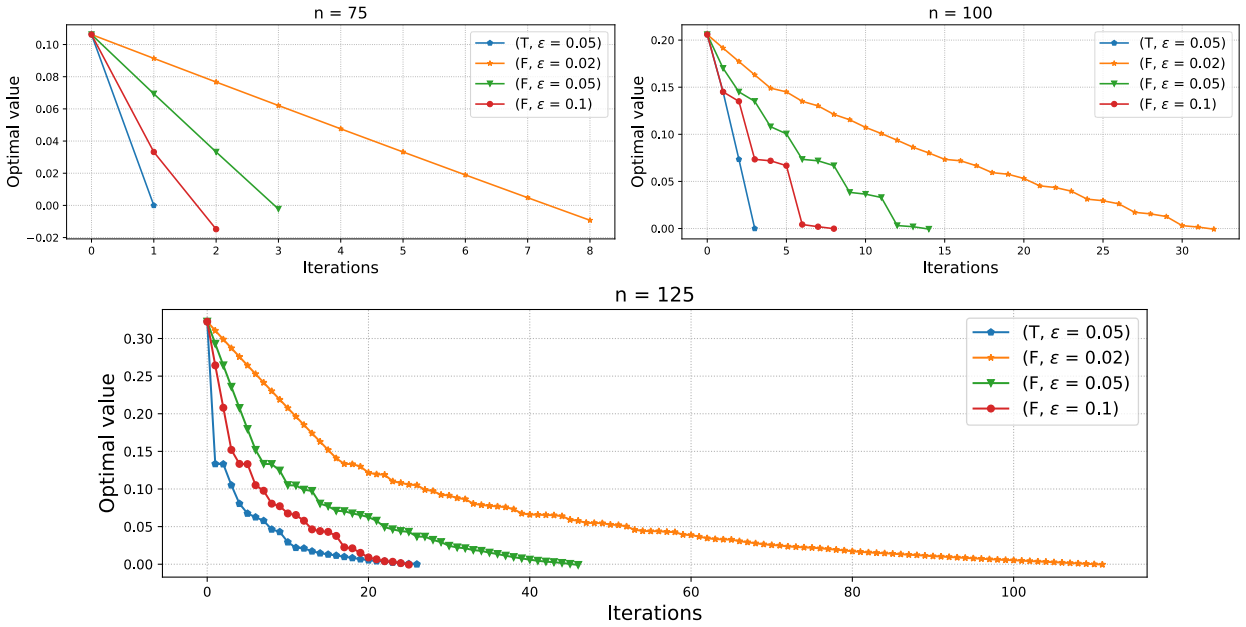

Figure 3: Decrease of the objective value of problem 11, for each n.
$p = 12\%, \alpha = 0.8$.

n	p	Algorithm and parameters	# of iterations	# of modified elements	Time [s]	$f(A)$	$g(A)$
75	0.1	`All`, (V = T, $\epsilon = 0.05$)	3	2	2.557	0.101	0.00356
		`All`, (V = F, $\epsilon = 0.02$)	15	5	12.231	0.059	0.00373
		`All`, (V = F, $\epsilon = 0.05$)	7	3	5.804	0.090	0.00400
		`All`, (V = F, $\epsilon = 0.1$)	5	3	4.002	0.120	0.00533
		`corrmat_h_weight`	/	277	5.232	0.0347	0.00366
	0.15	`All`, (V = T, $\epsilon = 0.05$)	5	2	3.493	0.058	0.00200
		`All`, (V = F, $\epsilon = 0.02$)	12	2	8.444	0.058	0.00206
		`All`, (V = F, $\epsilon = 0.05$)	6	3	4.308	0.055	0.00234
		`All`, (V = F, $\epsilon = 0.1$)	4	2	3.079	0.100	0.00281
		`corrmat_h_weight`	/	416	6.363	0.0169	0.00251
	0.2	`All`, (V = T, $\epsilon = 0.05$)	64	29	63.771	3.488	0.0344
		`All`, (V = F, $\epsilon = 0.02$)	245	34	249.486	2.646	0.0327
		`All`, (V = F, $\epsilon = 0.05$)	100	28	91.768	2.805	0.0332
		`All`, (V = F, $\epsilon = 0.1$)	51	26	46.897	2.759	0.0335
		`corrmat_h_weight`	/	555	5.266	1.019	0.0605
	0.25	`All`, (V = T, $\epsilon = 0.05$)	314	107	248.760	10.005	0.1002
		`All`, (V = F, $\epsilon = 0.02$)	933	114	779.216	9.513	0.0995
		`All`, (V = F, $\epsilon = 0.05$)	375	106	305.992	9.630	0.0998
		`All`, (V = F, $\epsilon = 0.1$)	190	98	151.699	9.899	0.1009
		`corrmat_h_weight`	/	693	1.377	4.238	0.146

Table 2: Results obtained when $n = 75$, $\alpha = 0.8$.

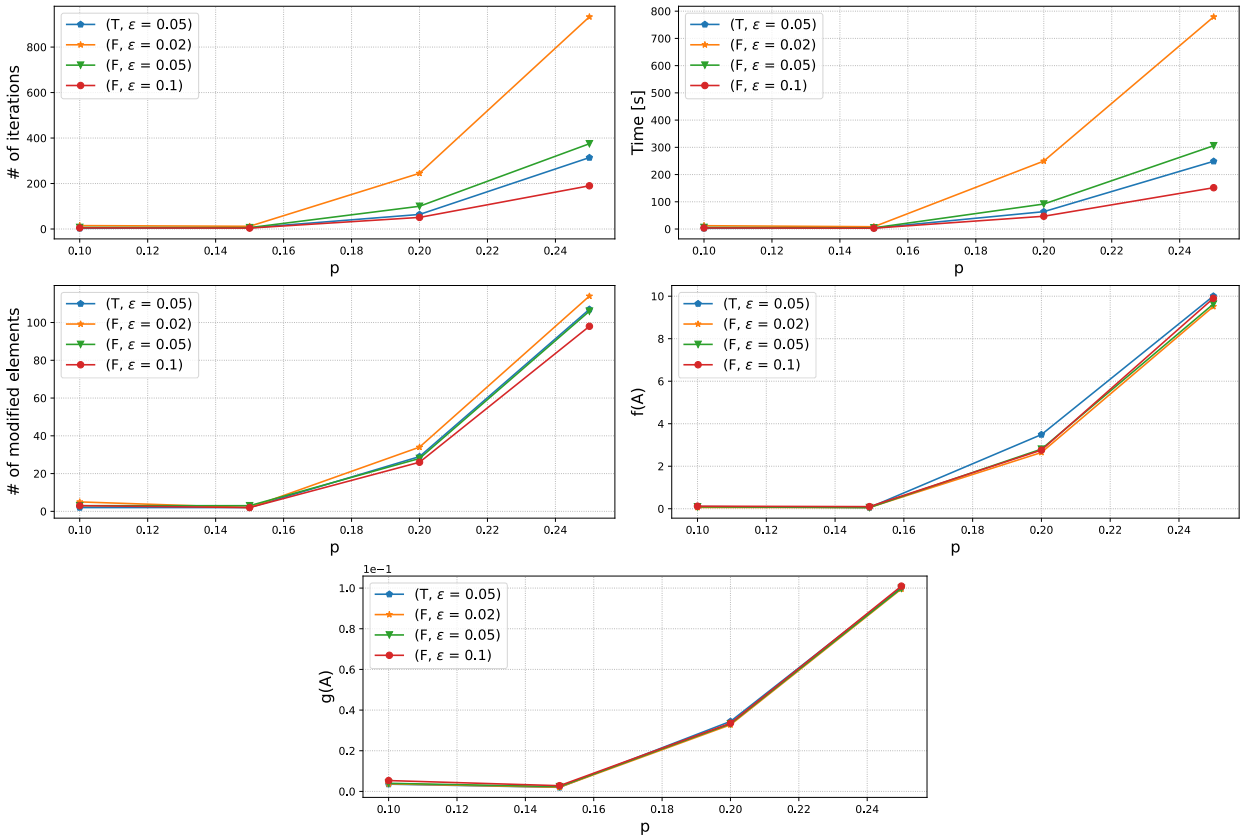

Figure 4: Results obtained with algorithm `All`, $n = 75$, $\alpha = 0.8$.

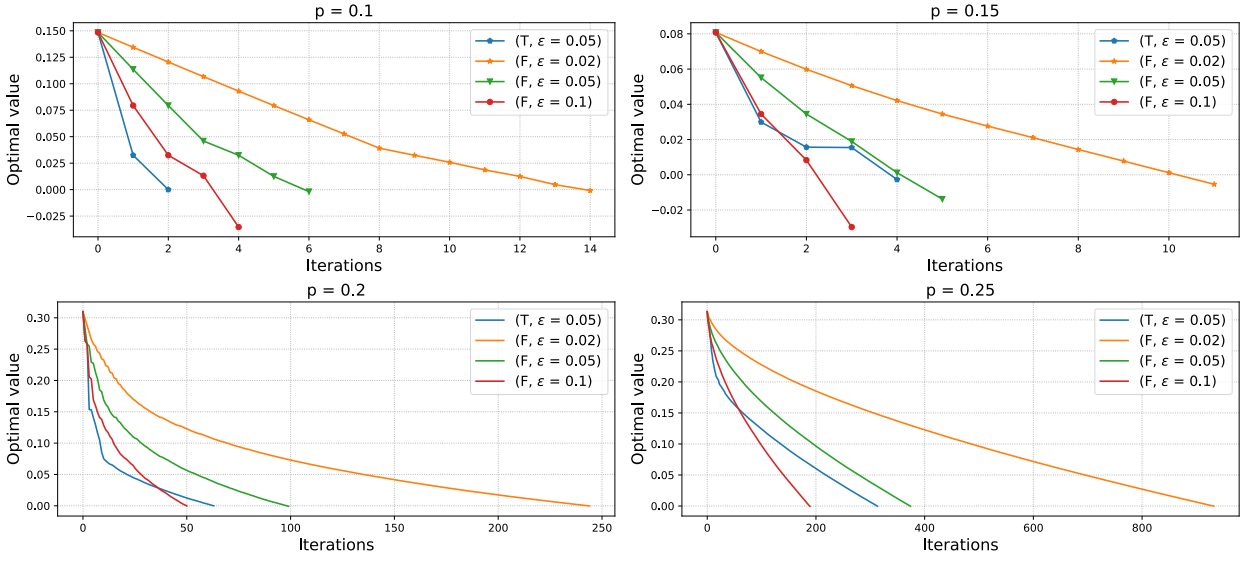

Figure 5: Decrease of the objective value of problem 11, for each p.
$n = 75, \alpha = 0.8$.

We analyze now the results obtained for the partial matrix provided by BNP. The table 3 contains the numerical results obtained and the graph on figure 6 represents the evolution of the value obtained for the function `SolveProblemOpti` for the different strategies. First of all, we do not consider an ϵ larger than 0.05 here because it causes too big changes and does not converge. The reason is explained in subsection 2.6. The size of the set $\bar{\Omega}$ is equal to 413 (165+248), which corresponds to a percentage p equal to 0.0183. The number of modified elements and the values of $f(A)$ and $g(A)$ are quite similar for the three different strategies, but the number of iterations is significantly bigger when $\epsilon = 0.02$. We observe in figure 6 that using the strategy (V = T, $\epsilon = 0.05$) results in a large decrease of the optimal value during the first iterations.

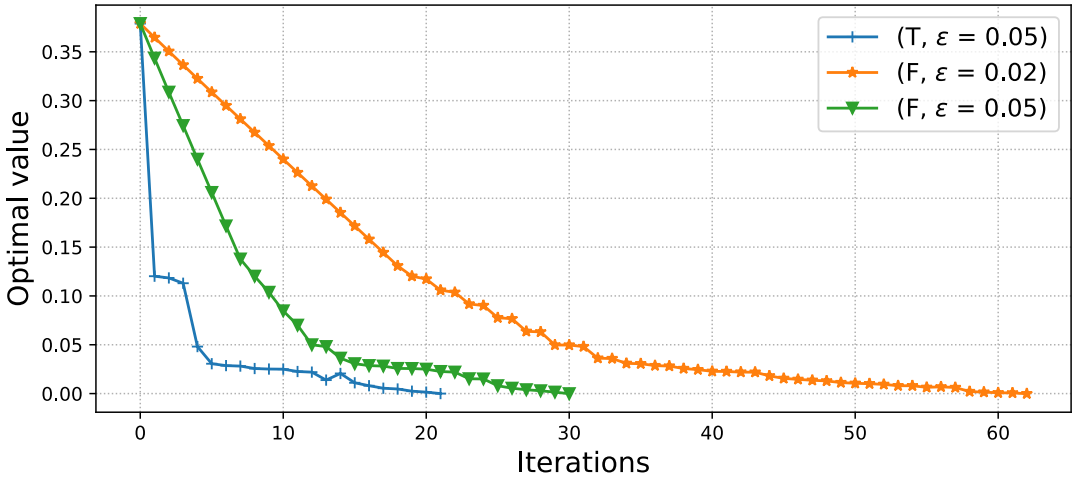

Figure 6: Decrease of the objective value of problem 11 with the matrix from BNP as input.

n	p	Algorithm and parameters	# of iterations	# of modified elements	Time [s]	$f(A)$	$g(A)$
165	0.0183	`All`, (V = T, $\epsilon = 0.05$)	22	18	1973	0.501	0.01000
		`All`, (V = F, $\epsilon = 0.02$)	63	20	5394	0.401	0.00896
		`All`, (V = F, $\epsilon = 0.05$)	31	17	2517	0.500	0.01084
		`corrmat_h_weight`	/	223	31	0.253	0.00896

Table 3: Results obtained with the matrix from BNP as input.

It is important to realize that there is no guarantee that algorithm `All` performs similarly when n and p correspond to one of the cases tested above. Indeed, the behavior of the algorithm depends strongly on which elements in A are fixed and on the values of these elements. For example, the number of elements fixed in the partial matrix from BNP is very low compared to the different cases tested above. It is very likely that if we construct the partial matrix from D (equation 31), with a percentage $p = 0.0183$ and $\alpha = 0.8$, that the resulting partial matrix will admit a semidefinite completion without any modification. The principal reason for this difference comes from the fact that the elements in the matrix provided are not fixed randomly. This particular structure is exploited in section 3. However, the tests made in this section allow to compare the behavior of the algorithm for the different parameters and to compare the results obtained with the ones from algorithm `corrmat_h_weight`.

2.6 Convergence

One natural question is "Does algorithm `Al1` always converge ?". Unfortunately, the answer is no. Indeed, we saw in inequation 23 that there was no guarantee of decrease in the objective value when we update the partial matrix A. For example, it may happen that the algorithm gets stuck in a situation where a same element in A is updated twice in a row with different signs. To clearly understand this, let us consider the following example of partial matrix :

$$A = \begin{pmatrix} 1 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & - \\ \frac{-1}{\sqrt{2}} & 1 & 0.05 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0.05 & 1 & \frac{-1}{\sqrt{2}} \\ - & \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 1 \end{pmatrix} \tag{34}$$

It is straightforward to see that the partial matrix does not admit a semidefinite completion since:

$$\det(A_{\{1,2,3\}}) = \det \begin{pmatrix} 1 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & 1 & 0.05 \\ \frac{1}{\sqrt{2}} & 0.05 & 1 \end{pmatrix} = -0.0525. \tag{35}$$

Let $\epsilon = 0.1$ and let's assume that the largest dual variable in absolute value is associated to the element $A_{23} = 0.05$ and is negative. After the update, the new partial matrix is :

$$A = \begin{pmatrix} 1 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & - \\ \frac{-1}{\sqrt{2}} & 1 & -0.05 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -0.05 & 1 & \frac{-1}{\sqrt{2}} \\ - & \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 1 \end{pmatrix} \tag{36}$$

Now, we have that :

$$\det(A_{\{2,3,4\}}) = \det \begin{pmatrix} 1 & -0.05 & \frac{-1}{\sqrt{2}} \\ -0.05 & 1 & \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 1 \end{pmatrix} = -0.0525 \tag{37}$$

Now, if the largest dual variable in absolute value corresponds to the element $A_{23} = -0.05$ and is positive, we are back with the same partial matrix as before. This will result in an infinite number of iterations. Of course, the assumption that the largest dual variable corresponds to the element A_{23} is false here, but for more complicated matrices, such phenomenon might happen. It happens for example with the partial matrix A provided by BNP and with the parameters (V = F, $\epsilon = 0.1$).

Note that in order to solve the problem, we check at each iteration in the algorithm if the partial matrix is the same as before and if it is the case, we update another element in A than the one corresponding to the largest dual variable.

2.7 Conclusion

In conclusion for this first approach, we developed a new algorithm that is able to solve the problem 4. The algorithm is strongly based on a semidefinite optimization problem (problem 9) that we have to solve several times. The dual variables of this problem (obtained by solving problem 11) give the optimal off-diagonal element in the partial matrix that requires modification. We developed two different approaches to modify the optimal element. The first approach consists of decreasing/increasing its value by a constant factor ϵ while the second approach (heuristic) is based on the specified principal minors of the partial matrix that are negative. We saw that all the execution time of the algorithm comes from the resolution of the optimization problem, which can be very slow for large partial matrices. This is the main drawback of the algorithm. In terms of results obtained, the number of modified elements is always small compared to the total number of fixed elements in the partial matrix. Moreover, the value of the function $g(A)$ is in the same order of magnitude as the one obtained with algorithm `corrmat_h_weight`, which indicates that the partial matrix is not too much modified. The use of the heuristic and the value of ϵ impact significantly the number of iterations in the algorithm. Finally, the convergence of the algorithm is not guaranteed but there exists a simple technique to solve the problem (see subsection 2.6). With this technique, we have never obtained a partial matrix for which algorithm `A11` had failed.

3 Second approach : Chordal graph

As we have seen in section 2, the initial approach, which involves iteratively solving the semidefinite optimization problem, performs well but tends to be slow for large partial matrices. In this new chapter, we delve into another approach to solve problem 4. This new approach is based on a theorem about chordal graphs, presented in subsection 3.2. In the upcoming subsection, we introduce and define carefully the new concepts that will be used to understand and prove the theorem.

3.1 Preliminaries

This section is strongly inspired by [16] and [17]. All the definitions and propositions come from these references. We incorporated a few additional remarks into the proof of proposition 4 to enhance its clarity.

Definition 5 (Graph). *A graph G is a pair $G = (V, E)$, where V is a set of nodes (vertices) and E is a set of edges. The set E is a subset of $\{(x, y) : x, y \in V\}$. The graph is undirected if $(x, y) \in E$ implies $(y, x) \in E$ and vice versa. The degree of a node is the number of edges connected to it.*

Definition 6 (Loop). *A loop is an edge that connects a node to itself: $(x, x) \in E$ is a loop for $x \in V$.*

Definition 7 (Clique). *A clique is a subset of nodes $C \subseteq V$ such that $(x, y) \in E,\ \forall\ x, y \in C$.*

Definition 8 (Cycle). *A cycle in G is a sequence of pairwise distinct nodes $\gamma = (v_1, v_2, ..., v_s)$ having the property that $(v_1, v_2), (v_2, v_3), ..., (v_{s-1}, v_s), (v_s, v_1) \in E$. s is referred as the length of the cycle.*

Definition 9 (Chord). *A chord of the cycle γ is an edge $(v_i, v_j) \in E$ where $1 \leq i < j \leq s, (i, j) \neq (1, s)$ and $|i - j| \geq 2$. In other words, a cycle has a chord if there is a pair of vertices that are adjacent, but not along the cycle.*

For example, the edge (v_3, v_{11}) represented in figure 7 represents a chord for the cycle $\gamma = (v_0, v_1, ..., v_{11})$.

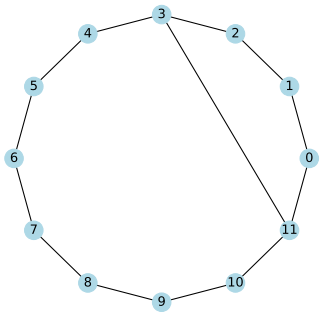

Figure 7: A cycle with chord.

Definition 10 (Minimal cycle). *The cycle γ is minimal if γ has no chord.*

Definition 11 (Induced subgraph). *An induced subgraph of $G = (V, E)$ is a graph $G' = (V', E')$, where $V' \subseteq V$ and $E' = \{(x, y) \in E : x, y \in V'\}$.*

Definition 12 (Chordal graph)**.** *A graph is chordal if there are no minimal cycles of length* ≥ 4.

For example, removing the edge (v_2, v_4) in the graph represented in figure 8 would result in a minimal cycle (v_1, v_2, v_3, v_4) of length 4 and hence to a non chordal graph.

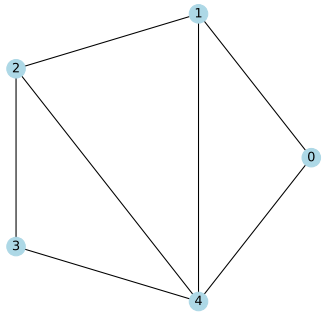

Figure 8: Example of chordal graph.

Definition 13 (Pattern of a matrix)**.** *The pattern of a partially specified matrix* A *is defined as the set of positions corresponding to the specified entries of the matrix. If* A *is an* $n \times n$ *partially specified matrix, its pattern can be represented by a graph* $G = (V, E)$*, where :*

$$V = \{1, ..., n\},$$
$$E = \{(i, j) : A_{ij} \text{ is specified }, i, j \in V\}.$$

Example : The partial matrix A in equation 38 is associated with the graph in figure 9.

$$A = \begin{pmatrix} 1 & 0.5 & - & 0.7 \\ 0.5 & 1 & 0.6 & 0.8 \\ - & 0.6 & - & - \\ 0.7 & 0.8 & - & 1 \end{pmatrix}. \tag{38}$$

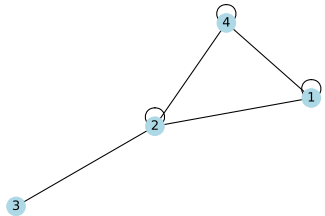

Figure 9: Pattern of the matrix defined in 38.

Note that the set of nodes $C = \{2, 3\}$ is not a clique of the graph since A_{33} is not specified.

Definition 14 (G-partial matrix)**.** *A* G*-partial matrix is a partially specified matrix associated with the pattern represented by the graph* G.

Definition 15 (Completion)**.** *A completion of the G-partial matrix A with $G = (V, E)$ is a $n \times n$ matrix M which satisfies $M_{ij} = A_{ij}$ $\forall (i,j) \in E$. M is a positive (semi)definite completion of A if and only if M is a completion of A and M is positive (semi)definite.*

The next two definitions are crucial and frequently referenced throughout the rest of this section.

Definition 16 (G-partial positive (semi)definite)**.** *Let A be a G-partial matrix. We say that A is a G-partial positive (semi)definite matrix if*

$$A_{ij} = A_{ji} \quad \forall (i,j) \in E$$

and for any clique C of G, the principal submatrix $[A_{ij} : i, j \in C]$ of A is positive (semi)definite .

Definition 17 (Completable/nonnegative-completable)**.** *We say that the graph G is completable (nonnegative-completable) if and only if any G-partial positive (semi)definite matrix has a positive (semi)definite completion.*

We conclude this section by showing that the terms "completable" and "nonnegative-completable" coincide. This allows us to only consider one case and only use one term in our notations.

Define L to be the set of vertices of G that have loops, i.e. $x \in L \iff (x,x) \in E$. We assume without loss of generality that $L = \emptyset$ or $L = \{1, ..., k\}$ for some $1 \leq k \leq n$.

Proposition 3. *G is completable (nonnegative-completable) if and only if the graph on L induced by G is completable (nonnegative-completable).*

Proof. ("⇒") Let G' denote the graph induced on L. If G is completable, then G' is completable since any clique of G must be contained in L. Indeed, being G' completable means that any G'-partial positive definite matrix has a positive definite completion, which is guaranteed by the fact that any G-partial positive definite matrix has a positive definite completion.

("⇐") Now, G' is completable and we want to prove that G is also completable. To prove that, consider the general $n \times n$ matrix $A(x)$ defined in equation 39. The submatrix A_{11} is a $k \times k$ matrix and is the positive definite completion of a partial matrix with pattern G'. The matrix H is an $(n-k) \times (n-k)$ arbitrary symmetric matrix and the matrix A_{12} is an $k \times (n-k)$ arbitrary matrix.

$$A(x) = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & x \times I + H \end{pmatrix}. \tag{39}$$

Since H and A_{12} are arbitrary, the matrix $A(x)$ can represent any matrix with pattern G. Then it is sufficient to remark that $A(x)$ is positive definite for sufficiently large positive x, which implies that G is also completable.

The arguments of the proof stay the same when G is nonnegative-completable. □

In view of proposition 3, we will therefore assume that $L = \{1, ..., n\}$, i.e. that all the diagonal entries are specified.

Proposition 4. *G is completable if and only if G is nonnegative-completable.*

Proof. ("⇒") Assume that G is completable and that A is a G-partial positive semidefinite matrix. Let $A_k = A + (1/k)I$, so that A_k is a G-partial positive definite matrix for each $k = 1, 2, ...$ Let M_k be a positive definite completion of A_k (M_k ∃ since G is completable). Since $L = V$, the sequence $\{M_k\}$ is bounded. Indeed, $L = V$ implies that the matrix A has all its diagonal elements that are specified and hence finite. This implies that every entries of the matrix are bounded for each k. Indeed, imagine by contradiction that the entry $M_{k_{ij}}$ is not bounded. It implies that the determinant of the submatrix $M_{k_{\{i,j\}}}$ is necessary negative which is impossible by theorem 5 since M_k is positive definite. The fact that each entry of M_k is bounded for all k implies that the sequence $\{M_k\}$ is bounded and has a convergent subsequence (by Bolzano Weierstrass theorem). The limit of this subsequence will then be a positive semidefinite completion of A.

("⇐") Assume that G is nonnegative completable and that A is a G-partial positive definite matrix. Choose $\epsilon > 0$ such that $A - \epsilon I$ is still a G-partial positive definite matrix. Let M be a positive semidefinite completion of $A - \epsilon I$. Then $M + \epsilon I$ is a positive definite completion of A. Hence, G is completable. □

In view of proposition 4, we only use the term "completable".

3.2 Theory about chordal graphs

In subsection 3.1, we defined several graph theory concepts that are used in this subsection to prove the main theorem, which establishes a connection between chordal graphs and the notion of completability. We proceed first with a sequence of lemmas preliminary to the proof of this theorem. Again, the proof of each lemma comes from [16] and [17]. We incorporated a few additional remarks into the proof of lemma 3 to enhance its clarity.

Lemma 1. *$G = (V, E)$ has no minimal cycle of length exactly 4 if and only if the following holds :*

For any pair of vertices u and v with $u \neq v, (u, v) \notin E$, the graph $G + (u, v)$ has a unique maximal clique which contains both u and v. (That is: if C and C' are both cliques in $G + (u, v)$ which contain u and v, then so is $C \cup C'$).

Proof. ("⇒") Assume G has no minimal cycle of length 4 and that u, v, C, C' are as described above. We will show that $C \cup C'$ is a clique in $G + (u, v)$. Let $z \in C, z' \in C'$; we show that (z, z') is an edge of $G + (u, v)$.

- If $z = z'$: This is trivial.
- If $(z, z') \cap (u, v) \neq \emptyset$: if for example $u = z$, we have that u and z' are both vertices in C' and hence (z, z') is an edge of $G + (u, v)$. The cases $v = z, v = z'$ and $u = z'$ lead to the same conclusion.
- The last case is when u, v, z, z' are distinct : It is easy to see that (z, u, z', v) is a cycle of G, since $(z, u), (v, z) \in C$ and $(u, z'), (z', v) \in C'$. However, G has, by hypotheses, no minimal cycles of length 4. Therefore, either (u, v) or (z, z') must be an edge of G. Since (u, v) is not an edge in G, it implies that (z, z') is an edge of G and therefore also an edge of $G + (u, v)$.

So in all the cases, $C \cup C'$ is a clique in $G + (u, v)$.

("⇐") For the converse, assume that $\gamma = (x, u, y, v)$ is a minimal cycle of G, such that (x, y) and (u, v) are not edges of G. Then $C = \{x, u, v\}$ and $C' = \{y, u, v\}$ are cliques of $G + (u, v)$ which contain u and v, whereas $C \cup C'$ is not a clique of $G + (u, v)$, since (x, y) is not an edge of G or $G + (u, v)$. Therefore, G cannot have a minimal cycle of the length exactly 4, with $C \cup C'$, a clique in $G + (u, v)$. □

It is straightforward from the definition of chordal graphs that they must satisfy Lemma 1.

Lemma 2. *Let $G = (V, E)$ be chordal. Then there exists a sequence of chordal graphs*

$$G_i = (V, E_i), \quad i = 0, 1, ..., s$$

such that $G_0 = G$, G_s is the complete graph, and G_i is obtained by adding an edge to G_{i-1} for all $i = 1, ..., s$.

Proof. The proof of Lemma 2 requires also preliminary lemmas and is available in [17]. □

The example in figure 10 (taken from [17]) illustrates the lemma. We start with a graph $G = (V, E)$, with $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (2, 3), (3, 4), (2, 5), (3, 5)\}$. At each step and until we obtain the complete graph, an edge is added to the graph such that the resulting graph stays chordal. The choice of the edge to add is not arbitrary. For example, choosing the edge $(1, 4)$ instead of $(1, 3)$ would lead to a non chordal graph.

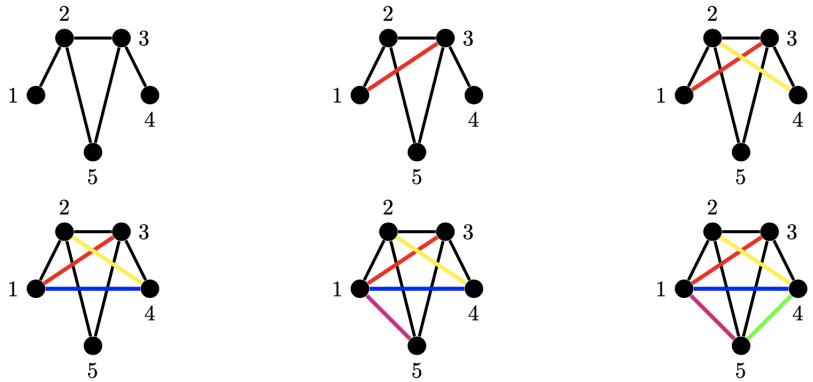

Figure 10: Example of sequence for lemma 2, ref. : [17]

Lemma 3. *Let $G' = (V', E')$ be a subgraph of $G = (V, E)$ induced by a subset $V' \subset V$. If G is completable, then so is G'.*

Proof. Let A' be any G'-partial positive semidefinite matrix. Define a G-partial positive semidefinite matrix A via :

$$A_{ij} = \begin{cases} A'_{ij} & \text{if } (i,j) \in E' \\ 0 & \text{if } (i,j) \in E \backslash E' \end{cases} .$$

G is indeed a G-partial positive semidefinite matrix since for any clique C of G, we must be in one of the two following cases :

- All nodes of C are in G'. In this case, the principal submatrix $[A_{ij} : i, j \in C]$ is positive semidefinite since A' is a G'-partial positive semidefinite matrix.
- One or several nodes of C are not in G'. In this case, let C' be the set of nodes in C that are contained in G'. The principal submatrix $[A_{ij} : i, j \in C]$ is then composed of another principal submatrix $[A_{ij} : i, j \in C']$ (which is positive semidefinite) and rows/columns of zeros for the nodes $\in C \backslash C'$. These rows/columns only create eigenvalues equal to zero, which implies that $[A_{ij} : i, j \in C]$ is also positive semidefinite.

Now, since G is completable, there is a positive semidefinite completion M of A. The principal submatrix M' of M corresponding to rows and columns indexed by V' is then a positive semidefinite completion of A'. □

Lemma 4. *There is a unique $k \times k$ positive semidefinite matrix A which satisfies :*

$$A_{ij} = 1, \quad \text{for all } |i-j| \leq 1. \tag{40}$$

Namely, A is the matrix of all 1's.

Proof. We only have to prove uniqueness since the existence is trivial. For $k = 2$, conditions in 40 characterizes entirely A and lemma 4 holds. For $k = 3$, we obtain :

$$A = \begin{pmatrix} 1 & 1 & - \\ 1 & 1 & 1 \\ - & 1 & 1 \end{pmatrix}.$$

Now, since all the principal submatrices have to be positive semidefinite, the only valid solution is to have "-" equal to 1. Indeed :

$$\det \begin{pmatrix} 1 & 1 & x \\ 1 & 1 & 1 \\ x & 1 & 1 \end{pmatrix} = -x^2 + 2x - 1 \geq 0 \iff x = 1.$$

Assume then than $k \geq 4$ and that the $k \times k$ matrix A is positive definite and satisfies conditions in 40. Let $1 \leq i \leq j \leq k$, $|i-j| \geq 2$. The 3×3 principal submatrix of A corresponding to rows and columns $i, i+1, j$ is positive semidefinite and satisfies conditions in 40. Hence $A_{ij} = 1$, since Lemma 4 holds when $k = 3$.
For example, let us consider the case $k = 4$, we have :

$$A = \begin{pmatrix} 1 & 1 & - & - \\ 1 & 1 & 1 & - \\ - & 1 & 1 & 1 \\ - & - & 1 & 1 \end{pmatrix}.$$

Now, if we consider the case $i = 1, j = 3$, the 3×3 principal submatrix A_{sub} corresponding to $i, i+1, j$ gives :

$$A_{\text{sub}} = \begin{pmatrix} 1 & 1 & - \\ 1 & 1 & 1 \\ - & 1 & 1 \end{pmatrix}$$

which imposes "-" to be equal to 1 as before. We obtain the exact same thing if we consider $i = 2, j = 4$ and $i = 1, j = 4$. At the end, the only possible solution is to have :

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

□

We are now ready to prove the main theorem :

Theorem 6. *The graph $G = (V, E)$ is completable if and only if the graph G is chordal.*

Proof. ("⇒") Let G be a completable graph and assume by contradiction that G is not chordal, i.e. that G has a minimal cycle γ of length ≥ 4. We may assume without loss of generality that $\gamma = (1, 2, ..., k)$. Let G' be the graph on $\{1, ..., k\}$ induced by G. Consider the $k \times k$ G'-partial matrix defined by :

$$A'_{ij} = 1 \quad \text{if} \quad |i - j| \leq 1$$
$$A'_{1k} = A'_{k1} = -1.$$

It is easy to see that A' is a G'-partial positive semidefinite matrix (we only need to check principal minors of order 2). By Lemma 4, A' is not completable to a positive semidefinite matrix, and so G' is not completable. However, G' is completable by Lemma 3, and we have a contradiction.

("⇐") Assume that G is chordal and is not the complete graph on $\{1, ..., n\}$. Let

$$G = G_0, G_1, ..., G_s$$

be a sequence of chordal graphs satisfying Lemma 2. Let A be any G-partial positive definite matrix. If we can show that there exists a G_1 partial positive definite matrix A_1 which extends A, then the existence of a positive definite completion of A will follow by induction. To have more intuition about the proof, consider the graphs G_0 and G_1 represented respectively on the left and right graphs of figure 11 and let A be the following partial matrix with pattern G_0 :

$$A = \begin{pmatrix} 1 & 1 & - & - & - \\ 1 & 1 & 1 & - & 1 \\ - & 1 & 1 & 1 & 1 \\ - & - & 1 & 1 & - \\ - & 1 & 1 & - & 1 \end{pmatrix}.$$

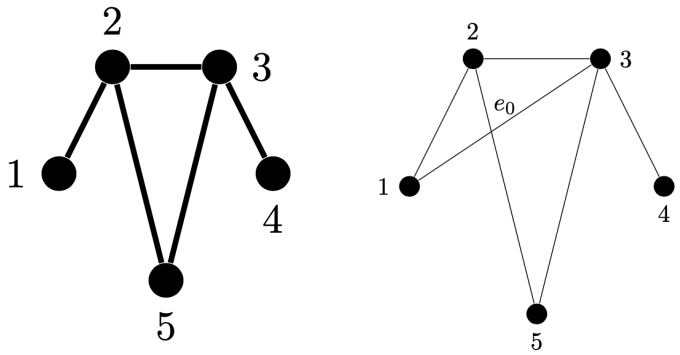

Figure 11: **Left** : G_0, **Right** : G_1
Loops are not represented. Taken from [17]

.

Denote by (u, v) the edge of G_1 that is not an edge of G. By Lemma 1, there is a unique maximal clique C of G_1 which contains both u and v. We may assume without loss of generality that $C = \{1, ..., p\}$ and that $u = 1, v = p$. In the example, the edge added is e_0, the clique $C = \{1, 2, 3\}$ and $(u, v) = (1, 3)$.

For any real number z, let $A_1(z)$ denote the G_1-partial matrix extending A which has a z in position $(1, p)$ and $(p, 1)$. Let $M(z)$ be the leading principal $p \times p$ submatrix of $A_1(z)$. In the example, we have :

$$A_1(z) = \begin{pmatrix} 1 & 1 & z & - & - \\ 1 & 1 & 1 & - & 1 \\ z & 1 & 1 & 1 & 1 \\ - & - & 1 & 1 & - \\ - & 1 & 1 & - & 1 \end{pmatrix} \qquad M(z) = \begin{pmatrix} 1 & 1 & z \\ 1 & 1 & 1 \\ z & 1 & 1 \end{pmatrix}.$$

Since any clique containing $(1, p)$ is a subset of $\{1, ..., p\}$, $A_1(z)$ is a G_1-partial positive definite matrix if and only if $M(z)$ is a positive definite matrix. Hence we need only to show that $M(z_0)$ is positive definite for some z_0. The graph on C induced by G has exactly the set of edges :

$$E_C = \{(i, j) : |i - j| \leq p - 2, 1 \leq i \leq j \leq p\}.$$

Therefore, the graph on C induced by G is a band graph, which is completable by theorem 1 of [16]. Hence, there exists a value z_0 as required. In the example, the only solution is $z = 1$ (by lemma 4). □

Before concluding this section, let us see two examples to clearly understand the theorem 6.

Example 1 :

Let A be the partial matrix defined in equation 41. We want to know if there exists a positive semidefinite completion of matrix A.

$$A = \begin{pmatrix} 1 & 0.3 & - & - & - & 0.6 & - & - & - \\ 0.3 & 1 & 0.5 & - & - & 0.6 & - & - & - \\ - & 0.5 & 1 & -0.2 & 0.4 & 0.3 & - & - & - \\ - & - & -0.2 & 1 & 0.3 & - & - & - & - \\ - & - & 0.4 & 0.3 & 1 & -0.1 & - & - & - \\ 0.6 & 0.6 & 0.3 & - & -0.1 & 1 & 0.4 & - & 0.5 \\ - & - & - & - & - & 0.4 & 1 & 0.3 & 0.15 \\ - & - & - & - & - & - & 0.3 & 1 & 0.4 \\ - & - & - & - & - & 0.5 & 0.15 & 0.4 & 1 \end{pmatrix} \tag{41}$$

The graph associated to the pattern of A is represented in figure 12. It is straightforward to see that the graph has no minimal cycle of length ≥ 4 and hence is chordal. Before applying theorem 6, we have to check that A is a G-partial positive semidefinite matrix, i.e. that for any clique C of G, the principal submatrix $[A_{ij} : i, j \in C]$ is positive semidefinite. One can see that it is the case for the matrix A considered. Hence, by theorem 6, the matrix has a positive semidefinite completion.

Example 2 :

Now, consider the following partial matrix A :

$$A = \begin{pmatrix} 1 & 0.9 & - & -0.9 \\ 0.9 & 1 & 0.9 & - \\ - & 0.9 & 1 & 0.9 \\ -0.9 & - & 0.9 & 1 \end{pmatrix}. \tag{42}$$

The graph of the partial matrix is represented in figure 13. It is easy to see that for any clique C of the graph, the principal submatrix $[A_{ij} : i, j \in C]$ is positive semidefinite. Indeed, the only cliques are $\{0, 1\}$,$\{1, 2\}$,$\{2, 3\}$ and $\{0, 3\}$, and they all lead to positive semidefinite principal submatrices. However, since the graph is not chordal, we cannot use theorem 6 to conclude whether or not a semidefinite completion exists. In fact there exists no positive semidefinite completion of this matrix.

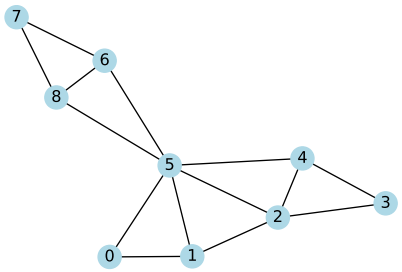

Figure 12: Graph associated to the partial matrix A defined in 41.
The loops are not represented.

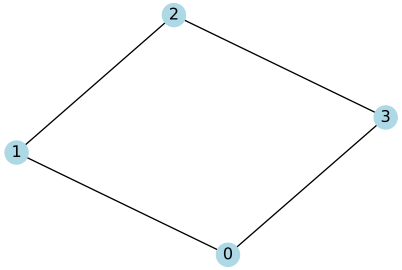

Figure 13: Graph associated to the partial matrix A defined in 42.
The loops are not represented.

The goal of the second approach is to exploit theorem 6, i.e. to exploit the fact that if the graph of the partial matrix A is chordal and if for any clique C of the graph, the principal submatrix $[A_{ij} : i, j \in C]$ is positive semidefinite then a semidefinite completion necessarily exists. Before presenting the algorithm, we first need to introduce the notion of clique tree.

3.3 Theory about clique Tree

This subsection presents the concept of clique trees, which is a new approach to represent chordal graphs. This subsection is inspired by [18].

Definition 18 (Tree)**.** *A tree in graph theory is an undirected graph such that any two nodes are connected by exactly one path.*

Definition 19 (Clique Tree)**.** *A clique tree of a graph $G = (V, E)$ is a tree T which has as nodes, the cliques of G. A clique tree T has the induced subtree property if for every $v \in V$, the cliques that contain v form a subtree (connected subgraph) of T.*

Example : The graphs on figure 14 represent a chordal graph and its clique tree representation. The example is taken from [19]. We observe that the clique tree representation has the induced subtree property. This is in fact guaranteed by theorem 7.

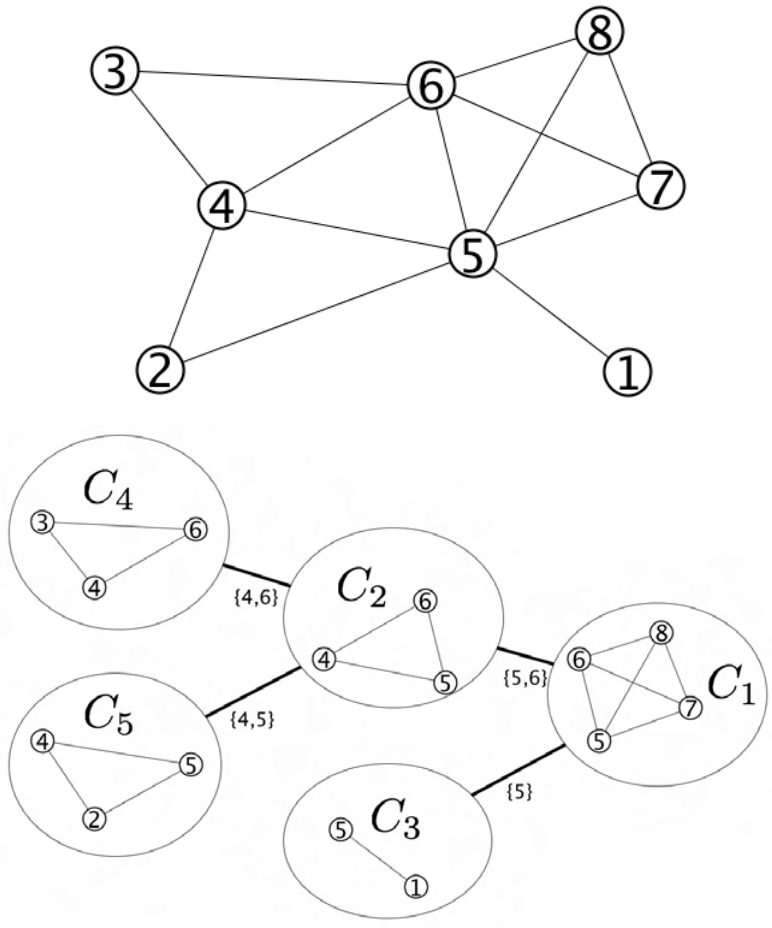

Figure 14: A chordal graph and its clique tree representation.
Source : [19]

.

Theorem 7. *Every chordal graph has a clique tree representation that has the induced subtree property.*

Proof. See theorem 3.5 in [18] □

It is worth noting that an algorithm exists to generate a clique tree representation of a chordal graph that satisfies the induced subtree property. The algorithm has a complexity quadratic in the number of maximal cliques of the graph [20]. A maximal clique is a clique that is not a subset of a larger clique. This last remark may appear to be insignificant but is of crucial importance.

Indeed, the number of cliques in a random graph quickly becomes very large, even for small n, while the number of maximal cliques generally evolves with a smaller rate. Consider for example the complete graph. It is well known that the number of cliques of a complete graph of n nodes is equal to $2^n - 1$. Indeed, since a clique can be represented by the nodes selected, the total number of cliques in the complete graph is the total number of possible subsets of the set $\{1, ..., n\}$, minus 1. We retrieve 1 because the empty set is not a clique. However, the number of maximal cliques in the complete graph is equal to one, i.e. the graph itself. This example is extreme, but illustrates the large difference that we can have between the total number of cliques and the number of maximal cliques. Moreover, if the graph is chordal, we have the following proposition :

Proposition 5. *A chordal graph with n nodes has at most n maximal cliques.*

Proof. See [21]. □

Therefore, since the clique tree representation of a chordal graph depends on the number of maximal cliques and not on the total number of cliques, the clique tree representation can be computed even for large graphs.

The objective of the next section is to present the algorithm developed to solve problem 4. The main idea of the algorithm is to use theorem 6. The first step is to transform the partial matrix such that the graph representing its pattern becomes chordal. The second step is to transform the partial matrix until each clique of the graph becomes associated with a positive semidefinite submatrix. If the two preceding steps are verified, theorem 6 guarantees that a positive semidefinite completion of the partial matrix exists. The clique tree representation of the chordal graph is used in the second step of the algorithm. It provides a structural way to explore all the cliques in the graph.

3.4 Algorithm

In this subsection, each step of the algorithm (called `Al2`) is presented in details with an example. The pseudocode for the algorithm is shown in 2. The input of the algorithm is a partial matrix A and the graph G corresponding to its pattern. Let us consider, as example, the following partial matrix A and its graph represented in figure 15.

$$A = \begin{pmatrix} 1 & - & - & - & -0.883 & - & - & - \\ - & 1 & - & 0.382 & -0.247 & - & - & - \\ - & - & 1 & 0.346 & - & -0.051 & - & - \\ - & 0.382 & 0.346 & 1 & - & - & - & - \\ -0.883 & -0.247 & - & - & 1 & 0.626 & -0.314 & -0.470 \\ - & - & -0.051 & - & 0.626 & 1 & 0.807 & 0.360 \\ - & - & - & - & -0.314 & 0.807 & 1 & 0.143 \\ - & - & - & - & -0.470 & 0.360 & 0.143 & 1 \end{pmatrix} \tag{43}$$

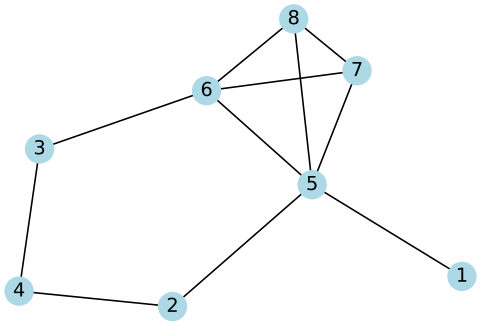

Figure 15: Graph associated to the partial matrix A defined in 43.

We assume that the graph G is connected, otherwise, the partial matrix A and the graph G are separated into connected components, and the algorithm is repeated for each component independently.

The first step of the algorithm is to make G chordal. There are different approaches to complete a graph in order to obtain a chordal one. A triangulation is the graph obtained when we add edges to a given graph to make it chordal. The approach used here is based on the minimum triangulation problem, which tries to find a triangulation of the graph with the fewest number of edges. Unfortunately, this problem is NP-hard [22], but there exists good heuristics such as the MCS-M algorithm. Instead of computing a minimum triangulation, the algorithm computes a minimal triangulation. A minimal triangulation H of a given graph G is such that no proper subgraph of H is a triangulation of G. The MCS-M algorithm has a time complexity in $\mathcal{O}(nm)$, where n is the number of nodes and m the number of edges. This is the best complexity known at this day for computing minimal triangulation of arbitrary graphs [23].

The MCS-M algorithm is implemented in the function `complete_to_chordal_graph` from Networkx. The minimal triangulation returned by the algorithm for the example considered is represented in figure 16.

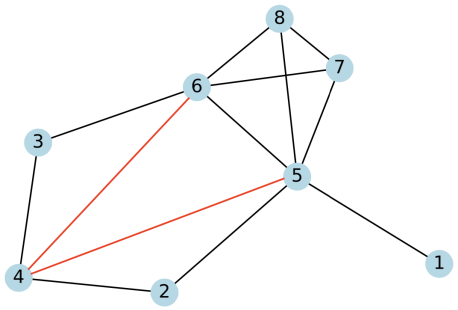

Figure 16: Minimal triangulation obtained with the MCS-M algorithm.

The second step of the algorithm is to obtain from the chordal graph its clique tree representation CT. The clique tree representation of the example considered is exactly the one represented in figure 14. The function `junction_tree` from Networkx is used for this part. The complexity of this second step is discussed in subsection 3.3.

The third step of the algorithm is to choose an initial root of the clique tree CT. Every node of CT can be chosen as initial root, but the choice can slightly impact the output obtained. Let's imagine that C_1 (see figure 14) is the initial root chosen. We then create an empty array called `NodesVisited` that will contain the nodes visited during the algorithm. We also create an empty array called `Conditions`. The role of this array will be clear in the next steps.

We start now an inner recursive function called `AnalyzeCliqueTree` that takes the partial matrix A, the graph CT, the initial root C_1, the array `Conditions` and the array `NodesVisited` as arguments.

In the first step of this inner function, we update the partial matrix A such that the principal submatrix associated with the initial root C_1, i.e. $[A_{ij} : i, j \in C_1]$, becomes positive semidefinite. In our example, the submatrix associated to C_1 is the following :

$$[A_{ij} : i, j \in C_1] = \begin{pmatrix} 1 & 0.626 & -0.314 & -0.470 \\ 0.626 & 1 & 0.807 & 0.360 \\ -0.314 & 0.807 & 1 & 0.143 \\ -0.470 & 0.360 & 0.143 & 1 \end{pmatrix} \tag{44}$$

The algorithm `Al1` with $\epsilon = 0.01$ and Variant = False (see section 2) is used to transform $[A_{ij} : i, j \in C_1]$ to a positive semidefinite matrix. We obtain :

$$[A_{ij} : i, j \in C_1] = \begin{pmatrix} 1 & 0.266 & -0.314 & -0.470 \\ 0.266 & 1 & 0.657 & 0.360 \\ -0.314 & 0.657 & 1 & 0.143 \\ -0.470 & 0.360 & 0.143 & 1 \end{pmatrix} \tag{45}$$

We also make the changes in A. Once this is done, we add the root C_1 to the array `NodesVisited` and check if there are any unvisited neighbors of C_1 in the clique tree graph CT.

For example, let us consider the node corresponding to the clique C_2 as a neighbor (see figure 14). In the graph G, the clique C_2 is composed by the nodes 4, 5, and 6, while the clique C_1 is composed by the nodes 5, 6, 7, and 8. Therefore, cliques C_1 and C_2 have the nodes 5 and 6 in common. The idea is to perform a recursion by invoking the function `AnalyzeCliqueTree` again, but this time with the root set as C_2 instead of C_1. The final objective is to enumerate all cliques in CT and ensure that the submatrix associated with each clique is positive semidefinite.

The remaining issue is that modifying the submatrix associated with clique C_2, i.e. $[A_{ij} : i, j \in C_2]$, could alter the value of A_{56}, which in turn could affect the positive semidefiniteness of the submatrix associated with clique C_1. To address this, we introduce the array `Conditions` and the function `Fix`. The purpose of the function `Fix` is to add, in the array `Conditions`, the elements in A that cannot be changed anymore, i.e. the elements of A that are common in both cliques.

For example, the only common element in C_1 and C_2 is A_{56} (and A_{65} by symmetry). Therefore, when the function `AnalyzeRoot` is called on C_2, we prevent `algorithm 1` from modifying the value of A_{56} (and A_{65}). In other words, the function `AnalyzeRoot` updates the partial matrix A in such a way that the submatrix associated with C_2 becomes positive semidefinite, while ensuring that A_{56} and A_{65} are fixed at the value of 0.266. Since A_{46} and A_{45} were not specified, their values are not fixed.

$$[A_{ij} : i, j \in C_2] = \begin{pmatrix} 1 & - & - \\ - & 1 & 0.266 \\ - & 0.266 & 1 \end{pmatrix}$$

Algorithm `Al1` with $\epsilon = 0.01$ and Variant = False, returns :

$$[A_{ij} : i, j \in C_2] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.266 \\ 0 & 0.266 & 1 \end{pmatrix}$$

Note that since the values of A_{56} and A_{65} cannot be modified, algorithm `Al1` finds the largest dual variable in absolute value associated to the partial matrix $[A_{ij} : i, j \in C_2]$, without taking into account the dual variable associated to the element A_{56} (and A_{65}).

The algorithm continues until the submatrices corresponding to all the cliques of CT become positive semidefinite. Note that the purpose of algorithm `Al2` is to use algorithms `Al1` several times but on much smaller matrices. Indeed, unless the partial matrix is fully specified, in which case algorithm `Al2` becomes equivalent to algorithm `Al1`, the algorithm `Al1` is applied on all the maximal cliques of G, which are smaller than the original matrix.

The convergence of algorithm `Al2` is discussed in subsection 3.5.

Algorithm 2:

Function Al2 (A, G);

Input : Partial matrix A and the graph G representing its pattern.
Output: New partial matrix $\bar{A}$ that admits a semi-definite completion.

```
G     ← MakeChordal(G)
CT    ← CliqueTree(G)
Root ← ChooseRoot(CT)
NodesVisited ← []
Conditions ← []

Function AnalyzeCliqueTree (A, CT, Root, Conditions, NodesVisited);

A ← AnalyzeRoot(A,Root,Conditions)
Add Root to NodesVisited
for Neighbor in neighbours(CT,Root) do
    if Neighbor not in NodesVisited then
        Conditions ← Fix(Conditions,Root,neighbor)
        Function AnalyzeCliqueTree (A, CT, Neighbor, Conditions , NodesVisited)
    end
end
Ā ← A
Return Ā
```

3.5 Proof of convergence

In this section, we demonstrate that under certain hypotheses, algorithm `Al2` always converges, meaning that it always returns a partial matrix that admits a positive semidefinite completion. Proposition 6 comes from [18] and is used in the proposition 7.

Proposition 6. *Let CT be a clique tree with the induced subtree property. Let C_1, C_2 be two nodes of CT. If $C_1 \cap C_2 \neq \emptyset$, then $C_1 \cap C_2$ is present in all the cliques that lie on the path in the clique tree between node C_1 and node C_2.*

Proof. For each vertex $v \in C_1 \cap C_2$, the cliques C_1 and C_2 are in the induced subtree that contains v. Therefore, all cliques on the path between C_1 and C_2 are also in the subtree. □

Proposition 7. *Each time the function `AnalyzeRoot` is invoked for a given clique, it is always possible to transform the submatrix associated with that clique into a positive semidefinite submatrix that satisfies the conditions specified in the array `Conditions`.*

Proof. It is trivial at the beginning of the algorithm since the array `Conditions` is empty. Suppose by contradiction that there exists a node C_2 in the clique tree CT such that its associated submatrix, i.e. $[A_{ij} : i, j \in C_2]$, cannot be transformed into a positive semidefinite submatrix. Since C_2 is not

the initial root, it implies that the function `AnalyzeRoot` was previously called with a neighbor of node C_2. Let us denote this node as C_1. Assume that in the graph G, clique C_1 contains the nodes $\{r_1, r_2, ..., r_k\}$. Without loss of generality, we can assume that clique C_2 contains the nodes $\{r_1, r_2, ..., r_l, s_1, ..., s_p\}$, where $1 \leq l \leq k$ and $p \geq 0$. It means that clique C_2 contains l nodes in common with clique C_1 and p new nodes. Let us assume, without loss of generality, that $p = 2$. The submatrix associated with C_2 can be expressed as follows :

$$[A_{ij} : i, j \in C_2] = \left[\begin{array}{cccc|cc} A_{r_1r_1} & A_{r_1r_2} & \cdots & A_{r_1r_l} & 0 & 0 \\ A_{r_2r_1} & A_{r_2r_2} & \cdots & A_{r_2r_l} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ A_{r_lr_1} & A_{r_lr_2} & \cdots & A_{r_lr_l} & 0 & 0 \\ \hline 0 & 0 & \ldots & 0 & 1 & 0 \\ 0 & 0 & \ldots & 0 & 0 & 1 \end{array}\right]. \tag{46}$$

It is straightforward to see that if the submatrix associated with C_1 is positive semidefinite then the matrix in equation 46 is also positive semidefinite. Note that the matrix can be written as above since there cannot be any conditions imposed on the values A_{r_i,s_j} with $i = 1, ..., l,\ j = 1, ..., p$ and on the values A_{s_i,s_j} with $i, j = 1, ..., p,\ i \neq j$. Indeed, let us imagine that a condition has been imposed on A_{r_i,s_j} by another clique (call it C_0) before in the execution of the algorithm. By proposition 6, the nodes r_i and s_j must be present in all the cliques that lie on the path between clique C_0 and clique C_2. Therefore r_i and s_j must be on the clique C_1, which is not the case. Hence, the matrix $[A_{ij} : i, j \in C_2]$ is a valid matrix and is positive semidefinite. □

Note that proposition 7 says that a positive semidefinite submatrix that satisfies the conditions in the array `Conditions` always exists but not that algorithm `Al1` always find one. This is a big difference. In section 2, we observed that algorithm `Al1` sometimes fails to converge when all the elements of the input partial matrix can be modified. Consequently, if certain elements in the matrix are fixed, the probability that the algorithm fails to converge is even higher. However, assuming that algorithm `Al1` always finds a solution, the following proposition holds:

Proposition 8. *Under the assumption that algorithm `Al1` always finds a solution, even when some elements in the matrix cannot be modified, algorithm `Al2` always converges.*

Proof. Indeed, under the stated assumption, each clique that is in the array `NodesVisited` is associated to a submatrix that is positive semidefinite. At the end of the algorithm, all the cliques of the original graph are included in `NodesVisited`. Since the original graph is chordal, according to theorem 6, a positive semidefinite completion of the partial matrix necessarily exists. □

Note that in practice, with the technique developed in subsection 2.6, it is very rare to be in the situation where the function `AnalyzeRoot` does not find a solution.

3.6 Pattern of the matrix provided

At the beginning of this chapter, we explained that the second approach was motivated by the fact that the pattern of the matrix provided has a particular property. At this point in the chapter, it should be clear that the property in question is that the graph representing the pattern is nearly chordal, meaning that only a few edges (less than 10%) are needed to make it chordal. The graph is represented in figure 17. It contains 165 nodes and 413 edges (the loops are not represented). The edges in red are the edges added to make the graph chordal. Only 20 edges are needed.

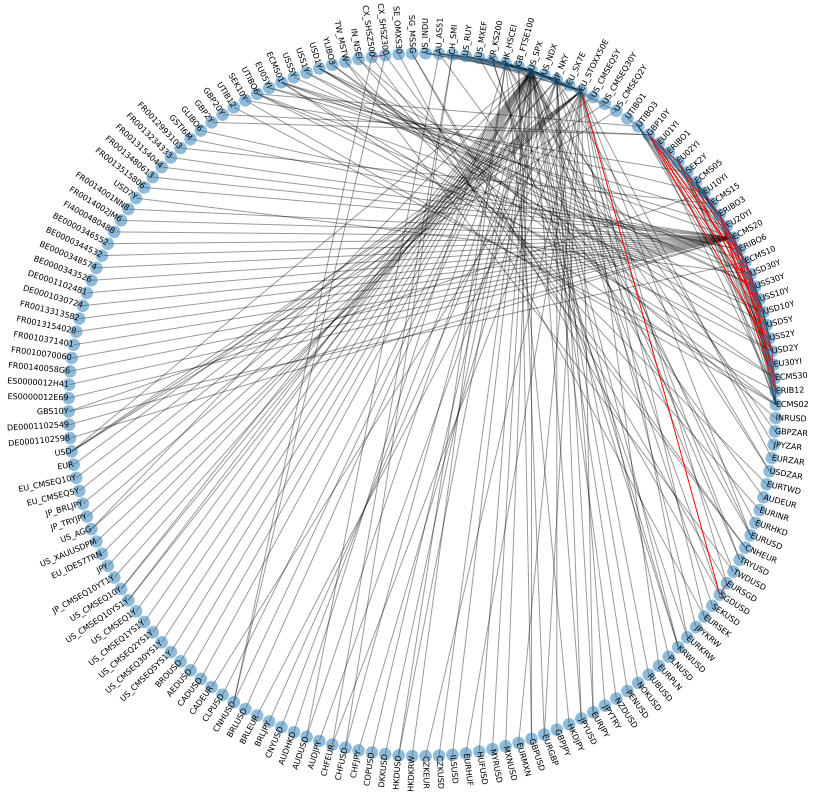

Figure 17: Pattern of the matrix from BNP.

It should be noted that, even though algorithm `Al2` can work with any partial matrix A, the algorithm is specifically designed to be efficient for partial matrices with chordal or nearly chordal graphs. In the case of a chordal graph, only the specified elements of the partial matrix are considered in the algorithm.

3.7 Results

In this subsection, the results obtained are presented. For each simulation, we compare the results obtained with algorithm `Al2` and with algorithm `Al1`. The two algorithms are compared on several partial matrices with different patterns. More precisely, a first comparison of the two algorithms is made when the inputs are partial matrices generated as in subsection 2.5, i.e. when the elements of the partial matrices are fixed randomly. A second comparison is made when the inputs are partial matrices with chordal patterns. The criteria used to compare the results are the same as in subsection 2.5.

First case : the elements are fixed randomly.

We start with the situation where the elements are fixed randomly. The parameter α in equation 31 is fixed to 0.7. The table in 4 summarizes the results obtained. The last column of the table indicates the number of additional edges required to make the graph representing the pattern chordal. The results are also shown graphically on figure 18. The results obtained with algorithm `Al2` are clearly worse than the ones obtained with algorithm `Al1`. Indeed, algorithm `Al2` modifies some elements of the partial matrix even when it admits a positive semidefinite completion. Moreover, algorithm `Al1` consistently yields improved results compared to algorithm `Al2`. The time taken by algorithm `Al1` is slightly longer for $n = 125$, but this difference is not significant considering the superior quality of the results obtained with algorithm `Al1`.

The principal reason of these very poor results is that for each n considered, the graph representing the patterns of the partial matrices are far from being chordal. Indeed, for each n considered, the number of edges in the graphs, given by $m = \frac{n(n-1)}{2}p$ (loops excluded), are approximately 4 times lower than the value m_c obtained.

Second case : the elements are not fixed randomly.

In the second case, the elements are fixed such that the graph representing the pattern of the partial matrix is chordal. We generate a chordal graph of n nodes and m edges as follows :

- We first generate a connected graph with n nodes and with $p(n-1)n/2$ edges (with $p = 0.1$).
- The second step is to make the graph chordal using the MCS-M algorithm. The problem is that the chordal graph obtained has a random number of edges m_f.
- The third step is to remove $m_f - m$ edges without losing the chordal property and without disconnecting the graph. At each step, we randomly select an edge and test whether the remaining graph remains chordal and connected. If it satisfies these conditions, the selected edge is removed. However, if the graph becomes non-chordal or disconnected, we randomly choose another edge and repeat the process.

Note that this procedure is quite slow but allows to obtain chordal graphs with a given number of nodes and edges. The edges in the chordal graph obtained give the positions of the fixed elements in the partial matrix A. Of course, the diagonal elements of A are also fixed to 1. The values of the elements in A are again generated with equation 31.

The numerical results obtained are written in table 5. We tested the behavior of algorithm `Al1`, algorithm `Al2` and algorithm `corrmat_h_weight` on partial matrices with patterns that are represented by chordal graphs of $n = 100$ nodes and $m = \{200, 400, 600, 800\}$ edges (loops excluded). Moreover, we tested the behavior of the algorithms when the parameter α in equation 31 varies. Note that algorithm `Al1` is always used with the parameter Variant equal to False and with an ϵ equal to 0.05.

Algorithm `Al2` is used with a parameter ϵ equal to 0.02 and algorithm `corrmat_h_weight` is parameterised as explained in subsection 2.5. The results obtained with algorithm `Al1` and algorithm `Al2` are also shown graphically in figure 19.

Based on the results obtained, we can make several comments :

- As expected, every criteria considered increase when m increases and when α increases.
- On the second graph of figure 19, we observe that the number of modified elements is generally higher with algorithm `Al2` than with algorithm `Al1`.
- On the first and fourth graphs of figure 19, we observe that the values of the functions $f(A)$ and $g(A)$ obtained are very similar for the two algorithms.
- On the third graph of figure 19, we observe that the execution times for algorithm `Al2` are significantly less important than the ones for algorithm 1. This is a massive improvement.
- In table 5, we observe that the algorithm `corrmat_h_weight` always leads to higher numbers of modified elements and to smaller values of $f(A)$. The values of $g(A)$ obtained are often slightly higher than the ones obtained with algorithm `Al1` and `Al2`.
- Increasing the value of α by 0.1 changes considerably the values obtained.

p	n	Algorithm and parameters	# of iterations	# of modified elements	Time [s]	$f(A)$	$g(A)$	m_c
0.15	50	`Al1`, $\epsilon = 0.05$	1	0	0.161	0.000	0.0000	450
		`Al2`, $\epsilon = 0.02$	/	10	0.556	0.172	0.0125	450
	75	`Al1`, $\epsilon = 0.05$	1	0	0.7179	0.000	0.0000	1324
		`Al2`, $\epsilon = 0.02$	/	71	3.106	0.736	0.0262	1324
	100	`Al1`, $\epsilon = 0.05$	5	1	18.514	0.080	0.0011	2896
		`Al2`, $\epsilon = 0.02$	/	133	22.396	5.639	0.0665	2896
	125	`Al1`, $\epsilon = 0.05$	18	7	269.787	0.285	0.0031	4858
		`Al2`, $\epsilon = 0.02$	/	238	106.489	13.755	0.1045	4858

Table 4: Results obtained when $p = 15\%$, $\alpha = 0.7$, random patterns.

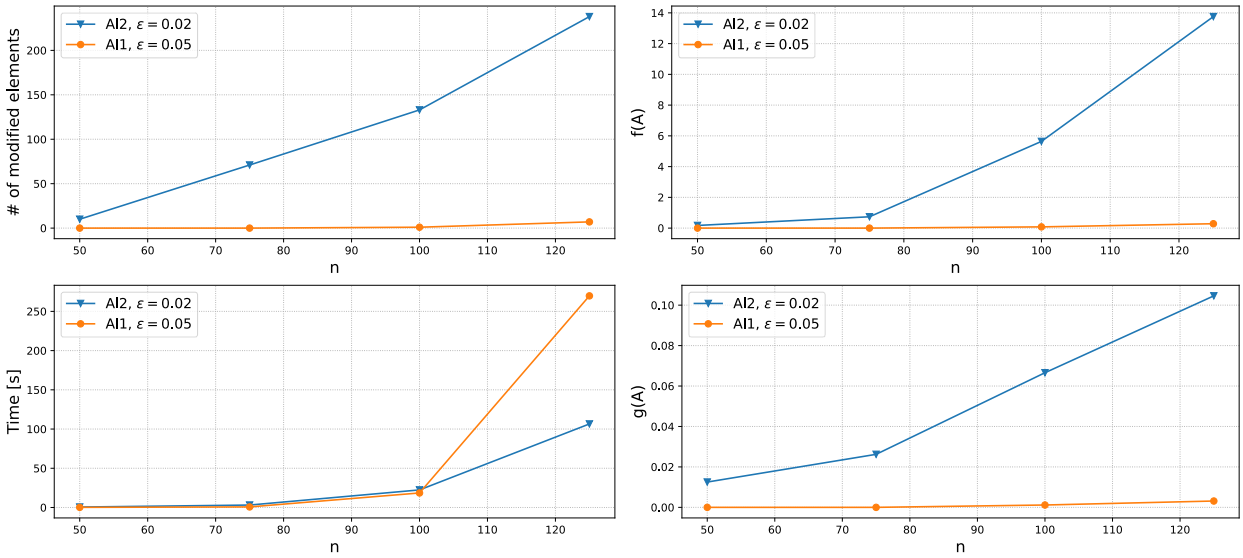

Figure 18: Results obtained when $p = 15\%$, $\alpha = 0.7$, random patterns.

n	α	m	Algorithm and parameter	# iterations Algo	# of modified elements	Time [s]	$f(A)$	$g(A)$
100	0.3	200	`Al1`, $\epsilon = 0.05$	1	0	3.754	0.0	0.0
			`Al2`, $\epsilon = 0.02$	/	0	0.126	0.0	0.0
			`corrmat_h_weight`	/	0	0.001	0.0	0.0
		400	`Al1`, $\epsilon = 0.05$	1	0	4.570	0.0	0.0
			`Al2`, $\epsilon = 0.02$	/	0	0.176	0.0	0.0
			`corrmat_h_weight`	/	356	0.186	1.7×10^{-10}	2.57×10^{-6}
		600	`Al1`, $\epsilon = 0.05$	10	6	35.9	0.0750	0.0075
			`Al2`, $\epsilon = 0.02$	/	10	0.75	0.1160	0.0103
			`corrmat_h_weight`	/	582	1.34	0.0047	0.0125
		800	`Al1`, $\epsilon = 0.05$	80	42	319	1.03	0.0484
			`Al2`, $\epsilon = 0.02$	/	63	4.84	1.14	0.0507
			`corrmat_h_weight`	/	795	1.13	0.139	0.0630
	0.4	200	`Al1`, $\epsilon = 0.05$	2	1	9.00	0.005	0.00175
			`Al2`, $\epsilon = 0.02$	/	2	0.136	0.0032	0.00140
			`corrmat_h_weight`	/	72	1.27	9×10^{-5}	0.00140
		400	`Al1`, $\epsilon = 0.05$	31	17	138	0.340	0.0270
			`Al2`, $\epsilon = 0.02$	/	21	1.10	0.439	0.0270
			`corrmat_h_weight`	/	396	0.87	0.054	0.0380
		600	`Al1`, $\epsilon = 0.05$	202	82	809	4.27	0.118
			`Al2`, $\epsilon = 0.02$	/	105	6.87	3.27	0.108
			`corrmat_h_weight`	/	598	0.648	0.921	0.126
		800	`Al1`, $\epsilon = 0.05$	1054	251	4168	4.94	0.158
			`Al2`, $\epsilon = 0.02$	/	229	19.5	5.26	0.152
			`corrmat_h_weight`	/	799	0.539	1.81	0.172
	0.5	200	`Al1`, $\epsilon = 0.05$	5	1	18.3	0.08	0.00598
			`Al2`, $\epsilon = 0.02$	/	9	0.174	0.05	0.00777
			`corrmat_h_weight`	/	181	0.887	0.008	0.0102
		400	`Al1`, $\epsilon = 0.05$	142	72	587	2.09	0.108
			`Al2`, $\epsilon = 0.02$	/	95	3.20	1.98	0.102
			`corrmat_h_weight`	/	399	0.670	0.66	0.114
		600	`Al1`, $\epsilon = 0.05$	754	241	3333	4.25	0.159
			`Al2`, $\epsilon = 0.02$	/	302	19.7	4.00	0.161
			`corrmat_h_weight`	/	599	0.817	1.83	0.173
		800	`Al1`, $\epsilon = 0.05$	1776	422	6944	8.25	0.213
			`Al2`, $\epsilon = 0.02$	/	433	43.4	8.40	0.206
			`corrmat_h_weight`	/	800	0.346	3.82	0.223

Table 5: Results obtained when $n = 100$, chordal patterns.

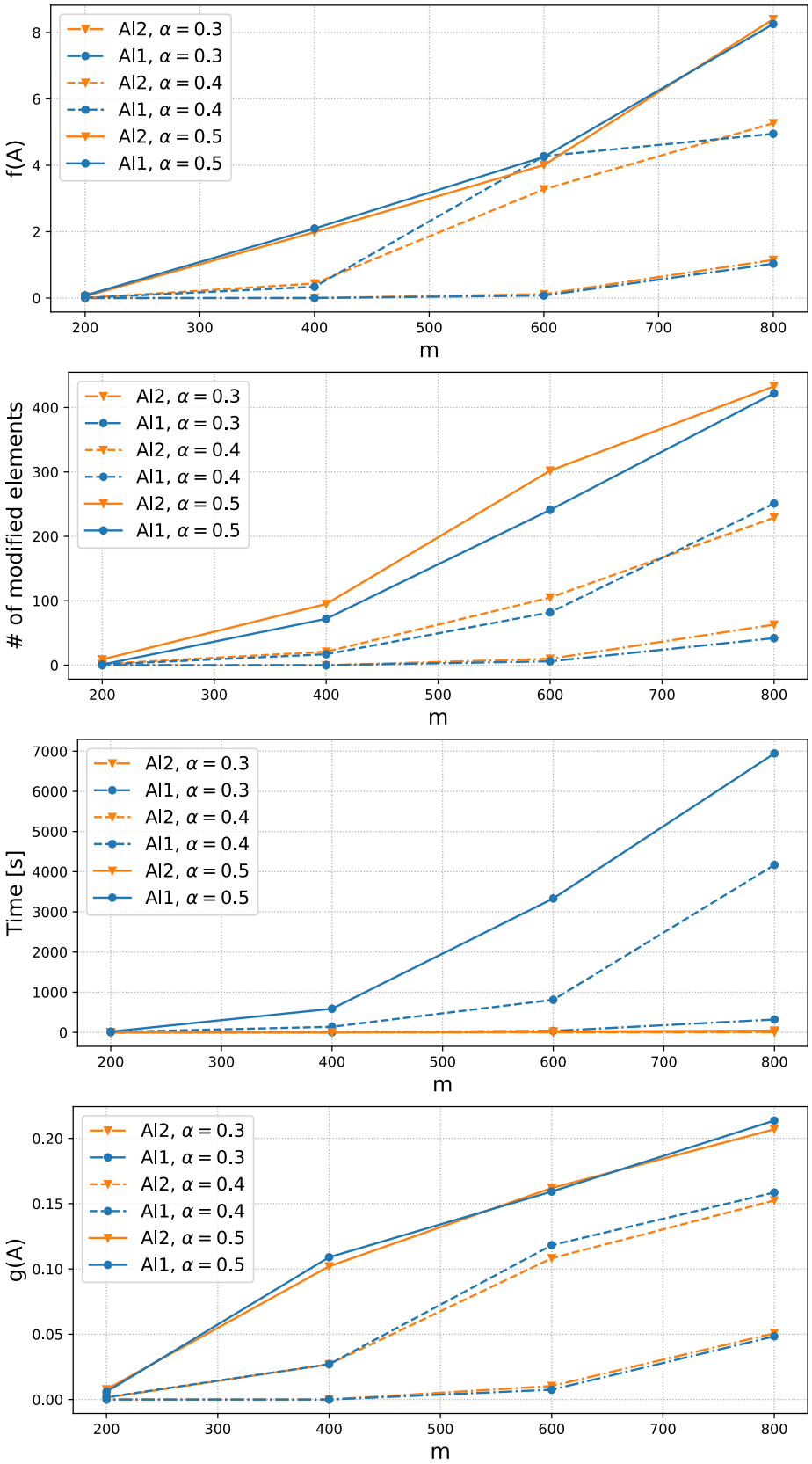

Figure 19: Results obtained when $n = 100$, chordal patterns

The numerical results for the matrix from BNP are presented in Table 6. Algorithm `Al2` was run with different values of ϵ. It is observed that smaller values of ϵ lead to a higher number of modified elements and to longer execution times, while yielding lower values of $f(A)$ and $g(A)$. By comparing these results with those obtained from algorithm `Al1` (Table 3), it can be noted that algorithm `Al2` performs slightly worse in terms of the results obtained but is significantly better in terms of execution times.

n	p	m	Algorithm and parameters ϵ	# of modified elements	Time [s]	$f(A)$	$g(A)$
165	0.0183	248	`Al2`, $\epsilon = 0.005$	29	0.728	0.691	0.01152
			`Al2`, $\epsilon = 0.01$	28	0.533	0.696	0.01178
			`Al2`, $\epsilon = 0.02$	27	0.454	0.727	0.01257
			`Al2`, $\epsilon = 0.05$	25	0.398	0.790	0.01445
			`corrmat_h_weight`	223	31	0.253	0.00896

Table 6: Numerical results obtained with the matrix from BNP as input.

In order to summarize the results obtained, we performed a last test to compare the three algorithms. The elements of the partial matrix A were generated with equation 31, with a parameter $\alpha = 0.5$ and a size $n = 100$. Let G be the graph representing the pattern of the partial matrix A. Let m_c denote the number of additional edges needed to obtain a chordal graph from G and let m be the number of edges in G (loops excluded). In the numerical experiment, the number m was fixed to 700. The graphs in figure 20 represent the results obtained when m_c/m increases. The factor m_c/m is a measure of "how far the graph is from being chordal". In other words, we tested the behaviors of the three algorithms when the graph representing the pattern of the partial matrix A, becomes less and less chordal.

We observe several things :

- When m_c/m is small, the values of $f(A)$, $g(A)$ and the numbers of modified elements are quite similar for algorithm `Al1` and `Al2`. However, the execution times are significantly lower for algorithm `Al2`.
- When m_c/m increases, the results obtained with algorithm `Al2` are clearly worse than the ones obtained with the other algorithms. This confirms the results obtained at the beginning of this subsection, i.e. where the elements of A were fixed randomly.
- Solving the problem becomes more challenging when the graph representing the pattern of the partial matrix is near to be chordal. Indeed, despite having the same number of fixed elements, it is evident from the graphs that the results obtained by the three algorithms progressively become better as the factor m_c/m increases. The principal reason is that, when the factor m_c/m increases, a small modification in the partial matrix A is less likely to generate errors in other parts of the matrix.
- Algorithm `Al1` always gives good results, i.e. is more robust than algorithm `Al2`, but is generally very slow for large partial matrices or when the pattern of the partial matrix becomes more complex.
- The algorithm `corrmat_h_norm` consistently modifies nearly all elements of the partial matrix. It achieves lower values of $f(A)$, but often results in higher values of $g(A)$. However, this algorithm is always fast for the partial matrices considered.

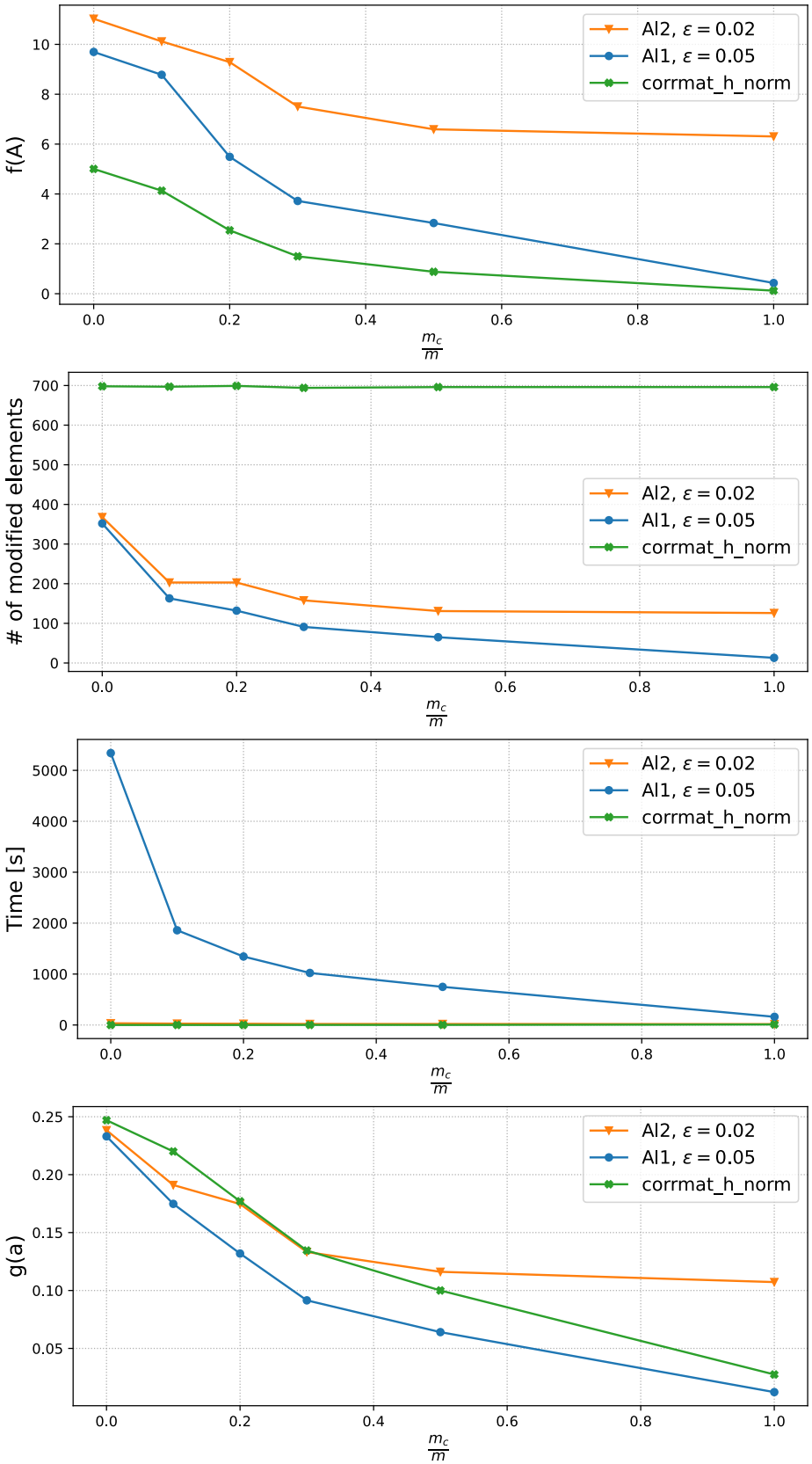

Figure 20: Results obtained when $n = 100$, $\alpha = 0.5$.

4 Conclusion

The objective of this master thesis was to solve the problem 4, i.e. to find an algorithm that transforms the partial matrix received such that a positive semidefinite completion exists. The algorithm should only make the necessary adjustments to the partial matrix without altering the values too drastically. Hence, there was a trade-off between the number of elements modified and the size of the modifications.

The first algorithm we developed (algorithm `Al1`) is strongly based on a semidefinite optimization problem (problem 11) that needs to be solved multiple times. The algorithm utilizes the dual variables associated with the semidefinite optimization problem to identify the optimal off-diagonal element in the partial matrix that requires modification. The use of dual variables allows for an efficient and effective selection of the specific element that should be adjusted, considering the constraints and the objective of the optimization problem. A first strategy is to decrease the value of the specified element chosen by a constant factor $\pm\epsilon$. The sign of ϵ is given by the sign of the dual variable corresponding to the specified element. We also developed an heuristic (see subsection 2.3) that uses a smarter approach to modify the elements in A. Instead of modifying each element by a constant value ϵ, we decrease, when possible, the specified element by a non constant value that is based on the specified principal minors in the partial matrix that are negative.

In terms of results, it was observed that, in general, the use of the heuristic resulted in a smaller number of modified elements, but resulted in a larger value of the function $f(A)$ (equation 32). The value of the function $g(A)$ (equation 33) was relatively independent of the use of the heuristic. We also observed that the value of ϵ has a significant impact on the number of iterations of the algorithm, i.e. on the number of times we have to solve the semidefinite optimization problem. A larger ϵ often leads to a smaller number of modified elements but to larger values of the functions $f(A)$ and $g(A)$. The first algorithm is robust, i.e. it works well for partial matrices that have different patterns. The main drawback of the algorithm is its execution time, which is directly proportional to the number of iterations needed to solve the problem. Since the cost of an iteration increases significantly with the size of the partial matrix, the method is not suitable for partial matrices with size larger than two-hundred.

In the second approach, we focused on the particular structure of the partial matrix provided by BNP to obtain a more efficient algorithm. We observed that the graph representing the partial matrix is almost chordal, i.e. we needed less than 10% of the number of edges to complete the graph to a chordal one. The algorithm that we developed (algorithm `Al2`) is mainly based on theorem 6 and on the clique tree representations of the chordal graphs. The main idea is to visit each clique of the graph and to transform the principal submatrix associated, to a positive semidefinite submatrix. Algorithm `Al1` is used inside algorithm `Al2` but is generally used on much smaller matrices! Note that algorithm `Al2` is equivalent to algorithm `Al1` when the partial matrix is fully specified.

Regarding the results, it is evident that algorithm `Al2` is not suited for partial matrices with graph representations that are far from being chordal. However, if the partial matrix has a graph that is near to be chordal, algorithm `Al2` gives generally similar or slightly worse results than algorithm `Al1` but takes significantly less time. Algorithm `Al2` has a tendency to modify more elements than algorithm `Al1` but often gives similar results in terms of the values of the functions $f(A)$ and $g(A)$.

For the partial matrix provided, the best results were achieved using algorithm `Al1` with the parameters $\epsilon = 0.02$ and variant = False. It is worth noting that the execution time for this configuration was 5394 seconds.

In conclusion, we recommend using algorithm `Al1` when precise and robust results are required. On the other hand, algorithm `Al2` is a good choice when the graph representing the pattern of the partial matrix is close to be chordal, and when the execution time is a critical factor.

References

[1] John C. Hull. *Options, futures, and other derivatives*. Pearson Prentice Hall, Upper Saddle River, NJ [u.a.], 6. ed., pearson internat. ed edition, 2006.

[2] Finance de marché. Qu'est-ce qu'un best-of call ? définition et payoff d'une option exotique sur panier.

[3] Nicholas Higham and Nataša Strabić. Anderson acceleration of the alternating projections method for computing the nearest correlation matrix. *Numerical Algorithms*, 72, 08 2016.

[4] Nicholas Higham, Natasa Strabic, and Vedran Sego. Restoring definiteness via shrinking, with an application to correlation matrices with a fixed block. *SIAM Review*, 58:245–263, 01 2016.

[5] Houduo Qi and Defeng Sun. A quadratically convergent newton method for computing the nearest correlation matrix. *SIAM J. Matrix Analysis Applications*, 28:360–385, 01 2006.

[6] Marianna E.-Nagy, Monique Laurent, and Antonios Varvitsiotis. Complexity of the positive semidefinite matrix completion problem with a rank constraint. *Mathematical Programming*, 03 2012.

[7] Kaifeng Jiang, Defeng Sun, and Kim-Chuan Toh. An inexact accelerated proximal gradient method for large scale linearly constrained convex sdp. *SIAM Journal on Optimization*, 22, 01 2012.

[8] Houduo Qi and Defeng Sun. An augmented lagrangian dual approach for the h-weighted nearest correlation matrix problem. *Ima Journal of Numerical Analysis - IMA J NUMER ANAL*, 30, 03 2010.

[9] Anupam Prakash, Jamie Sikora, Antonios Varvitsiotis, and Zhaohui Wei. Completely positive semidefinite rank. *Mathematical Programming*, 171(1-2):397–431, oct 2017.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[11] Bernd Gärtner and Jiří Matoušek. *Approximation Algorithms and Semidefinite Programming*. 11 2013.

[12] Fuzhen Zhang. Matrix theory. *Springer*, 2013.

[13] Erling Andersen, Cornelis Roos, and Tamás Terlaky. On implementing a primal-dual interior-point method for conic quadratic programming. *Mathematical Programming*, 95:249–277, 02 2003.

[14] MOSEK ApS. *MOSEK Optimizer API for Python 10.0.46*, 2019.

[15] The Numerical Algorithms Group. *The NAG Library for Python*.

[16] E.M. Sà R.Grone, C.R. Johnson and H.Wolkowicz. Positive definite completions of partial hermitian matrices. *Linear Algebra and its applications, 58*, pages 109–124, 1984.

[17] Hannah Entner. Matrix completion problems, 2020.

[18] Lieven Vandenberghe and Martin S. Andersen. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, 1(4):241–433, 2015.

[19] Romain Pogorelcnik. Décomposition par séparateurs minimaux complets et applications. 12 2012.

[20] Finn Verner Jensen and Frank Jensen. Optimal junction trees, 2013.

[21] Martin Charles Golumbic. The wonderful world of chordal graphs. *The 32nd European Conference on Combinatorial Optimization*, 2019.

[22] Mihalis Yannakakis. Computing the minimum fill-in is np-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.

[23] Anne Berry, Jean Blair, Pinar Heggernes, and Barry Peyton. Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39:287–298, 08 2004.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**